Advances in Volcanology

Alireza Hajian · Giuseppe Nunnari · Roohollah Kimiaefar

Intelligent Methods with Applications in Volcanology and Seismology

IAVCEI

Springer

Advances in Volcanology

An Official Book Series of the International Association of Volcanology and Chemistry of the Earth's Interior

Series Editor

Karoly Nemeth, Institute of Natural Resources, Massey University, Palmerston North, New Zealand

Advances in Volcanology is an official book series of the International Association of Volcanology and Chemistry of the Earth's Interior (IAVCEI). The aim of the book series is to publish scientific monographs on a varied array of topics or themes in volcanology. The series aims at building a varied library of reference works, by describing the current understanding of selected themes such as certain types of volcanism or regional aspects. The books in the series are prepared by leading experts actively working in the relevant field. The Advances in Volcanology series contains single and multi-authored books as well as edited volumes. The Series Editor, Dr. Karoly Nemeth (Massey University, New Zealand), is currently accepting proposals and a proposal document can be obtained from the Publisher, Dr. Annett Buettner (annett.buettner@springer.com).

Alireza Hajian · Giuseppe Nunnari ·
Roohollah Kimiaefar

Intelligent Methods with Applications in Volcanology and Seismology

Springer

Alireza Hajian
Department of Physics
Najafabad Branch
Islamic Azad University
Najafabad, Iran

Roohollah Kimiaefar
Department of Physics
Najafabad Branch
Islamic Azad University
Najafabad, Iran

Giuseppe Nunnari
Dipartimento di Ingegneria Elettrica,
Elettronica e Informatica
Università degli Studi di Catania
Catania, Italy

ISSN 2364-3277 ISSN 2364-3285 (electronic)
Advances in Volcanology
ISBN 978-3-031-15431-7 ISBN 978-3-031-15432-4 (eBook)
https://doi.org/10.1007/978-3-031-15432-4

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use. The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To Saint Fatima and Saint Agatha

Preface

When I was in Institute Nazionale di Geofisica e Vulcanologia (INGV) in Italy, Catania section, as a sabbatical for a period of about six months from 2018 to 2019, not only it was interesting for me but also especially a new experience for me to work on volcano geophysics data specially on lovely Etna, where various geophysical data are well gathered on a broad network of stations close to its craters. There, I got familiar with some of the geophysicists and computer scientists working hard on Etna data!

I believe "ETNA" is not only one of the best monitored active volcanos in the world but also is one of the biggest well-equipped geophysical Natural Labs of the world. Another thing I want to mention is that Etna is a very lovely beautiful mountain!! If you think I am false, please take a look to photos in Figs. 1, 2, 3, 4, 5. As another evidence of my claim, you can find by advanced search in virtual media and find that there are a lot of people (not essentially the experts) who love Etna and follow its activities with a high score of click per day!

As another example of the huge number of fans of Etna, I can mention my roommate, Claudio, at Catania, who was a geology undergraduate student and every night I met him in the kitchen while cooking was visiting Etna trends in discussion rooms and forums on Internet, it was very interesting and to some extent strange for me that he also check the records of tremors available at INGV, Catania website times and times every day. During my stay at Catania, Prof. Peter Styles came there to visit me for about 4 days and he was very eager to visit Etna volcano. He told me that he had been visiting Hawaiian volcano very close to the main craters and very eager to visit Etna closer, so with the help of Dr. Filippo Greco, the very kind and expert gravimetry scientist in INGV, we arranged a one-day field of visiting Etna for him. This shows clearly the interesting nature of Etna volcano not only for young researchers like me but also for prominent famous geophysicists (Fig. 5).

I met prominent scientists in Catania who highly improved my scientific life, especially Prof. Giuseppe Nunnari and Dr. Filippo Greco, who are men of ethics and brilliant scientists by all means. Giuseppe taught me a lot of things both in life and in artificial intelligent; how to work with machine learning and also the concepts and philosophy of deep learning; I learnt a lot from him. Also, Dr. Filippo Greco was who accepted and invited me to go Italy for a sabbatical period in his institute, INGV Catania section. He really helped me a lot, i.e., to settle in this city, to visit Etna geophysical stations

Fig. 1 Fuggy atmosphere of a close location to craters of Etna (*Photo by* A. Hajian, 2018)

Fig. 2 View of a silent day of Etna in a sunny day in Fall (*Photo by* A. Hajian, 2018)

Fig. 3 Snowy covered view of Etna around (*Photo by* A. Hajian, Winter 2018)

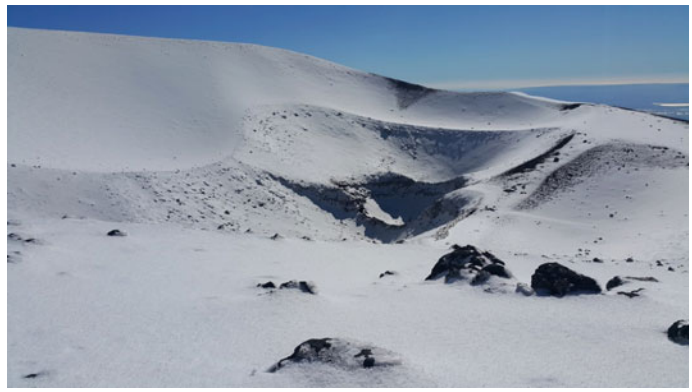


Fig. 4 Snow covered view of one of Etna craters (*Photo by* A. Hajian, Winter 2018)

Fig. 5 Field trip to Etna in a snowy day by Dr. Filippo Greco and Prof. Peter Styles (with very much thanks to Dr. Filippo Greco for arranging and coaching the field-trip, (*Photo by* A. Hjaian, 2018).)

close to its craters, very high-tech gravimeters that for me it was the first time to see and work with!, his office, preparing individual virtual machine for me to run my huge codes and data, infrastructure. Also, Dr. Flavio Cannavo, Associated Professor at INGV, helped me with both geophysical data of Etna and introducing the suitable indexes to evaluate classification quality of machine learning classifiers for Etna volcano activities based on integration of various geophysical time series. I had couple of discussion sessions with these kind talent scientists (Dr. Filippo, Prof. Guiseppe and Dr. Flavio) that I name these sessions at my memorial notes, during my stay at INGV Catania, as "**Golden**" sessions end with my "Persian Black Tea" most of the afternoons ready by my tiny electric tea maker joint with very nice "Italian Coffee: Espressos" but sooner than afternoon (Fig. 6)! Without the helps, experts and supports of them, I never could be success going deeper at my new field of the work at INGV!

Fig. 6 Drinking Persian Tea after a "Golden" discussion session on Machine learning for Etna!, at Dr. Filippo Greco's office at INGV, Catania, Italy, November 7, 2018 (Left to right: me, Dr. Flavio Cannavo, Dr. Filippo Greco, Prof. Giuseppe Nunnari)

When I came back to my country Iran, Persia, I taught a course for seismological Ph.D. students, namely "neural networks and applications in seismology," and during the semester while teaching this course based on my first Springer-Published book (Hajian and Styles, 2018), I deeply felt the lack of a technical text book for applications of intelligent methods in seismology; in the next semester when I was invited to teach a course, namely "advanced physics of the earth," my students motivated me to present my achievements in Italy about using machine learning for geophysical volcanology. Totally and finally, I decided to prepare a text book draft with detailed focus on the application of intelligent algorithms in both the seismology and volcanology. At a first glance, this was seemed as a very hard task to do alone, but I was sure that if the work was implemented through a team work it would be faster and easier in a Work Breaking Structure (WBS) framework. Then, I called Prof. Nunnari and discussed about it and invited him to help and guide me in this way; finally, he accepted to edit the whole text of the book and also to help in writing all the chapters with both the concepts and applications in seismology and volcanology. I think he is the most expert person in the world to do this as he has very high respected experiences of both working on Etna during his years of working in INGV and also on intelligent methods in Electrical Engineering Department at the University of Catania for more than 25 years. He has practically used different types of machine learning methods for various problems about Etna. However, I also invited Dr. Flavio Cannavo, who is a talent researcher of computer science with applications on volcanology and seismology, but most of the time, he is very busy with various research teams and projects. I am very indepted to him for both the data and scientific common sessions on comparison of his results by Bayesian to our results by machine learning about ongoing state of Etna volcano based on various time series of geophysical variations for a period of 5 years of Etna activity classes. Finally, one of my colleagues at Department of Physics, Dr. R. Kimiaefar, accepted my invitation to help as the third author of the book, especially with deep learning approaches.

This book is provided as a text book for both graduated students (M.Sc. and Ph.D.) and researchers who are interested to use intelligent methods as a tool for their problems in volcanology and/or seismology.

The book contains seven chapters, totally. In Chapter "Intelligent Methods and Motivations to Use in Volcanology and Seismology," we have presented an overall view of the intelligent methods and the motivations to apply them in volcanology and seismology. In Chapter "Machine Learning: The Concepts," the concepts of machine learning are introduced, while in Chapter "Machine Learning Applications in Volcanology and Seismology," we have explained various examples for application of machine learning in volcanology and seismology. In the next two chapters (Chapters "Deep Learning: The Concepts" and "Deep Learning: Applications in Seismology and Volcanology"), we have described deep learning and its applications in volcanology and seismology, respectively. Finally, in Chapter "Evolutionary Algorithms with Focus on Genetic Algorithm" Genetic Algorithm concepts and in the next chapter (Chapter "Application of Genetic Algorithm in Volcanology and Seismology"), its applications for volcanologists and seismologists are presented.

It is necessary to mention that each chapter is provided with some practical interesting examples to help the reader not only understanding the concepts of the intelligent methods but also helping him/her step-by-step as a guidance how to apply the method/s practically. Also, we did our best to present key codes and/or related toolboxes of designing, implementing and testing of the introduced intelligent methods in MATLAB. Some chapters, i.e., chapters "Deep Learning: The Concepts" and "Deep Learning: Applications in Seismology and Volcanology" are more based on Python coding which are more common for deep learning approaches.

Alireza Hajian (A.H.) would like to thank and appreciate Dr. Filippo Greco from INGV, Catania section, who supported him with his kind regards and infrastructure during his sabbatical leave period in Italy, also he expresses his utmost gratitude to his dear father Mohammad Hassan and his lovely kind mother Ozra, because he owes his entire existence to these two divine angels. Last but not least, A.H. extremely gratefuls to his darling wife Mohaddeseh and his lovely daughter Elina for the spare time that belonged to them but was spent writing the book over the course more than a year.

Najafabad, Iran — Assoc. Prof. Alireza Hajian
a.hajian@iaun.ac.ir

Catania, Italy — Prof. Giuseppe Nunnari
Najafabad, Iran — Asst. Prof. Roohollah Kimiaefar

Reference

Hajian A, Styles P (2018) Application of soft computing and intelligent methods in geophysics. Springer International Publishing AG, part of Springer Nature. https://doi.org/10.1007/978-3-319-66532-0

Contents

Intelligent Methods and Motivations to Use in Volcanology and Seismology 1
1 Introduction 1
1.1 Brief List of Intelligent Methods Applications in Volcanology and Seismology 4
1.2 Metaheuristic Algorithms 5
1.3 Intelligent Methods 6
1.4 The Role of Intelligent Methods Toward Big Volcano Science 13
References 15

Machine Learning: The Concepts 19
1 Introduction 19
2 An Overview of the State Estimation Problem 21
3 Parametric and Non-parametric Estimation of Densities 22
3.1 A Parametric Approach 22
3.2 Parametric Density Estimation: A Numerical Example 23
3.3 A Non-parametric Approach 24
3.4 Estimation of the Prior Probabilities 24
4 Supervised Classification 25
4.1 The Bayes Minimum Risk Classification 25
4.2 An Example of Bayesian Minimum Risk Classifier 27
4.3 Naive Bayes Classifiers 27
4.4 Parzen Classifiers 28
4.5 K-Nearest Neighbor (KNN) Classification 29
4.6 Classification Based on Discriminant Functions 30
4.7 The Support Vector Classifier 30
4.8 Decision Trees 32
4.9 Combining Models: Boosting and Bagging 34
4.10 Error-Correcting Output Codes (ECOC) 36
4.11 Hidden Markov Models 37
5 Classification Metrics for Model Validation 38
6 Unsupervised Classification 41
6.1 Hierarchical Clustering 41
6.2 K-Means Clustering 42

6.3 Fuzzy c-means . . . 43
6.4 Mixture of Gaussians . . . 44
7 Methods to Reduce the Dimensionality of a Dataset . . . 46
7.1 The Principal Component Analysis (PCA) . . . 47
7.2 Self-organizing Maps . . . 47
8 Software Tools for Machine Learning . . . 48
8.1 The MATLAB™ Statistical and Machine Learning Toolbox . . . 48
8.2 The Python Scikit-Learn Package . . . 48
8.3 The R Language . . . 49
8.4 The PRTools Library . . . 49
References . . . 49

Machine Learning Applications in Volcanology and Seismology . . . 51
1 Introduction . . . 51
2 ML to Classify Seismic Data . . . 51
3 Hidden Markov Model to Classify Volcanic Activity . . . 53
4 Earthquake Detection and Phase Picking . . . 55
5 Earthquake and Early Warning . . . 55
6 Ground Motion Prediction . . . 56
7 ML for Volcanic Activity Monitoring Based on Images . . . 57
8 Multi-parametric Approaches to Classify the Volcanic Activity . . . 59
9 Unsupervised Classification of Volcanic Activity . . . 63
10 Clustering Multivariate Geophysical Data by Using SOM . . . 65
References . . . 67

Deep Learning: The Concepts . . . 69
1 Introduction . . . 69
2 Deep Learning, an Overview . . . 69
3 Deep Learning, Pros and Cons . . . 71
4 Layers in a Deep Learning Models . . . 73
5 Deep Learning Models . . . 74
5.1 Supervised Deep Learning Methods . . . 74
5.2 Deep Convolutional Neural Network . . . 74
5.3 Image Classification by CNN: Fault Detection in Synthetic Seismic Data . . . 75
5.4 Recurrent Neural Networks . . . 81
5.5 Long Short Term Memory Network . . . 82
5.6 Gated Recurrent Unit Network . . . 82
5.7 Application of Long Short Term Memory Network for Extrapolating 2D Sequential Data . . . 84
5.8 Unsupervised Deep Learning Methods . . . 92
5.9 Unsupervised Auto Encoder Network . . . 92
5.10 Attenuating Random Noise in Gravity Data Using Auto Encoder Network . . . 92
References . . . 100

Deep Learning: Applications in Seismology and Volcanology 103
1 Introduction 103
2 Applications of Deep Learning in Seismology 103
2.1 Long-Range-Short-Term Earthquake Prediction Using CNN-BiLSTM-AM Model. 103
2.2 Real-Time Focal Mechanism Determination by Fully Convolutional Network 114
2.3 A Functional Very Deep Convolutional Neural Network Model for Fast and Precise Earthquake Phase Picking. 117
2.4 Magnitude Calculation Directly from Raw Waveforms Using MagNet 121
3 Detecting and Locating Induced Seismicity Using ConvNetQuake Model 123
3.1 Classification Based on Limited Training Samples Using CapsNet: Aplication to Microseismic Record Classification 124
3.2 Deep Convolutional Neural Network for Fast Prediction of Earthquake Intensity from Raw Accelerograms 126
4 Applications of Deep Learning in Volcanology 128
4.1 Volcano Deformation Identification Using Convolutional Neural Network Trained by InSAR Synthetic Database. 129
4.2 Automatic Classification of Volcano-Seismic Signals Using Active Deep Learning to Overcome the Case of Small Training Datasets. 135
4.3 Probabilistic Shape Classification of the Volcanic Ash Using Convolutional Neural Networks. 137
References. 139

Evolutionary Algorithms with Focus on Genetic Algorithm. 141
1 Evolutionary Computation 141
1.1 Introduction 141
1.2 Annals of Evolutionary Computing: A Brief Literature Review. 142
1.3 Biological and Artificial Evolutionary 142
2 Genetic Algorithm 142
2.1 Introduction to Natural Genetics 142
2.2 Crossover. 143
2.3 Mutation 143
2.4 Fitness 143
3 Fundamentals of Genetic Algorithms 143
3.1 Encoding 143
3.2 Fitness 143
3.3 Cross Over. 143
3.4 Mutation 144

4 How to Run Each Parts of the Genetic Algorithm? 147
4.1 Population Representation and Initializing the Algorithm 148
4.2 Objective Function and Fitting Function 150
4.3 Selection and Various Techniques 151
4.4 Different Techniques of Cross Over 153
4.5 Different Techniques of Mutation 153
5 Important Definitions in Running Genetic Algorithm 154
6 The General Process of Optimization and Problem Solving in Genetic Algorithms 155
7 More Examples on Operators in Genetic Algorithms 156
7.1 Encoding a Chromosome 156
7.2 Cross Over 158
7.3 Mutation 158
8 Investigation of Important Factors in Genetic Algorithm 158
8.1 Cross Over Rate 158
8.2 Mutation Rate 159
8.3 Population Size 159
8.4 Selection Methods 159
8.5 Different Types of Encoding 160
9 Genetic Algorithm in MATLAB; A Brief View 162
10 Random Numbers Generation in Matlab 164
10.1 Random Permutation 164
10.2 Pseudorandom Integers from a Uniform Discrete Distribution 165
10.3 Normally Distributed Pseudorandom Numbers 166
References 166

Application of Genetic Algorithm in Volcanology and Seismology 169
1 Introduction 169
2 Inverse Modelling of Volcanomagnetic Fields Using Genetic Algorithm 169
2.1 Mechanisms that Cause Volcanomagnetic Anomalies 169
2.2 Motivations to Use Genetic Algorithm for Inversion of Volcanomagnetic Anomalies 171
2.3 The GA Procedure to Invert Volcanomagnetic Anomalies 171
2.4 Forward Models 172
2.5 Testing and Evaluating the GA Performance for Synthetic Data 175
2.6 Evaluation of GA Results for Real Cases 175
3 Inversion of SAR Data in Active Volcanic Areas by Genetic Algorithm 176
3.1 Abstract 176
3.2 SAR Data Forward Modelling 177
3.3 Test of GA Method for Synthetic Data 178
3.4 Inversion of Real SAR Data 178

4 Automatic Monitoring System of Infrasonic Events at Mt. Etna Using Genetic Approach . . . 183
4.1 The Infrasonic Events Automatic Monitoring . . . 183
4.2 Genetic Algorithm Method for Infrasonic Source Parameters Estimation . . . 183
4.3 Evaluation of the Proposed Method . . . 185
5 Rapid Estimation of Earthquake Magnitude and Source Parameters Using Genetic Algorithms . . . 185
5.1 Displacement Detection and Estimation . . . 185
5.2 Moment Magnitude (Mw) Estimation . . . 187
6 Generator of Genetic Seismic Signals . . . 192
6.1 Introduction . . . 192
6.2 The Underlying Idea . . . 192
6.3 Evaluation of GA Seismic Generator . . . 193
7 Focal-Mechanism Determination in Taiwan by Genetic Algorithm . . . 194
7.1 The Study Region . . . 194
7.2 GA Procedure for Focal Mechanism Determination . . . 195
7.3 Test of GA for Synthetic Data . . . 195
7.4 Test for Real Data . . . 197
References . . . 202

Intelligent Methods and Motivations to Use in Volcanology and Seismology

Abstract

In this chapter we present a brief glance at the various kinds of intelligent methods and also their most important advantages are described. Furthermore the motivations to use intelligent methods in volcanology and seismology is explained with a briefed list of the recent applications.

1 Introduction

During the last decade, intelligent methods, methods based on Artificial Intelligent, have witnessed significant developments and have found many applications in science and technology such as geosciences, physics, chemistry, Computer science, Electrical and Computer Engineering, Mechanical Engineering, Civil Engineering, etc.

In the recent years, the interest to use some of intelligent and soft computing methods like "**Deep Learning**" and generally "**Machine Learning (ML)**" has been dramatically increased in the field of geophysics especially in seismic data processing, volcanology and seismology.

The "Intelligent Methods (IM)" refers to consortium of computational data-based methodologies. If we assume the IM as a tripod then its main triple bases are: Fuzzy Logic (FL), Neural Networks (NN) and Genetic Algorithm (GA), depicted in Fig. 1.

The scope of the intelligent methods is very wide so that from another point of view they can be divided into various branches and sub-branches as shown in Fig. 9. The progress is this field, especially in **deep learning,** is obviously seen through the increasing in the number of commercial and engineering software products in the market i.e. Python and Matlab specialized toolboxes developed for design and implement of **Machine Learning** algorithms.

This is important to notice that in today's highly integrated world of the science; most of the developments to extend the boundaries of the knowledge due to multi-interdisciplinary aspects. To get a comprehensive understanding about complex natural phenomena, i.e. volcanoes and earthquakes, there is a highly need to integrate a large amount of different types of spatial and/or temporal data and information, containing a lot of various recorded geophysical signals, geological information, geochemistry data, and meteorological data. Also, there are common fields of study for volcanology and seismology i.e. "**Volcanic earthquakes/Tsunamis**" (Fig. 2). Almost all of volcanic eruptions have some earthquakes activity beneath or close to the volcano crater/s whether before, during and after the paroxysm.

The movement of magma beneath the volcano's active volcanoes; most of the times lead to volcanic quakes (Fig. 3). Two main categories of

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

A. Hajian et al., *Intelligent Methods with Applications in Volcanology and Seismology*, Advances in Volcanology, https://doi.org/10.1007/978-3-031-15432-4_1

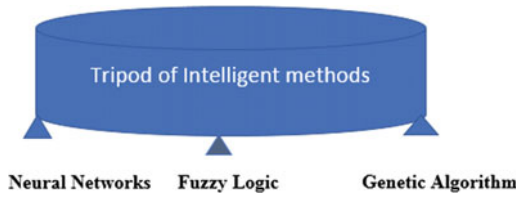


Fig. 1 Three main basses of intelligent methods

volcanic earthquake waves are "Long Periods" and "Swarms" (Fig. 4).

To discover a deeper knowledge of both volcanoes and earthquakes, as very complex systems, there are two main challenges: **First**, analyzing a large amount of different types of data and information and **Second** the uncertainty available on this big data, as most of the natural variables and parameters are imprecise or namely "fuzzy".

We can assume the "volcano" and/ or "earthquake" as a natural non- linear chaotic system and then, in order to understand/ recognize this system, try to apply a nonlinear system identification approach. In this state, the problem is that this system is not man-made and we need to the records of this system from its activity start up to now or at least for a long period of decades, but it's really impossible as the volcano or the fault, as sources of eruption and earthquakes respectively, are both very much older than that of human's life. This point of view might be crucially and disappointed. I personally think this is a pessimist look to the problem! A better and optimist look to this problem can be achieved if we think that forever the age of the volcano/ fault is very much more than the total years of our registered records but we are going to know its present or close future statement i.e. a doctor doesn't essentially need to a long period of records of a heart-attacked patient, however he/she might be very old.

The second mentioned problem is generally simply solved by "fuzzy thinking". In contrast to classical analyzing techniques that use crisp mathematics, fuzzy set theoretic techniques provide an elegant foundation and a large set of rich methodologies to analyze imprecise data in diverse geophysical data processing and more qualified near-to-real interpretations.

Our brain works like a "**fuzzy system**" because the data gathered by our five-senses are all "imprecise", "vague" or "fuzzy", but the brain output is so precise. For example, the brain doesn't know the environment (as external space) temperature and body as internal space while it controls the body temperature very precisely so that a healthy person body's temperature is 37 °C. If you ask a healthy person that how much the room temperature is, he/she only can anticipate it but can't tell you exactly while his/her brain is adjusting his/her body reference to the imprecise temperature measurement by the sense of touch. Working precisely, intelligently and adaptively with a good level of flexibility is the art of the human brain, if we understand this art deeper and deeper then we can implement the method to be more successful to a deeper understanding of volcano/fault to get to better interpretation from available "imprecise" data. The key to discover this brain amazing the art of the work is its "**fuzzy logic**" based framework or its "**fuzzy system**" working structure.

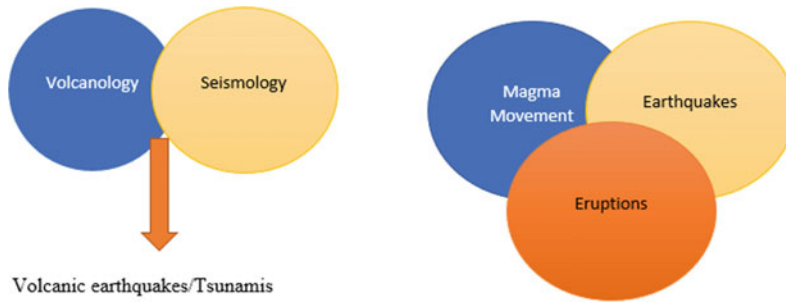


Fig. 2 A schematic of the common fields of volcanology and seismology

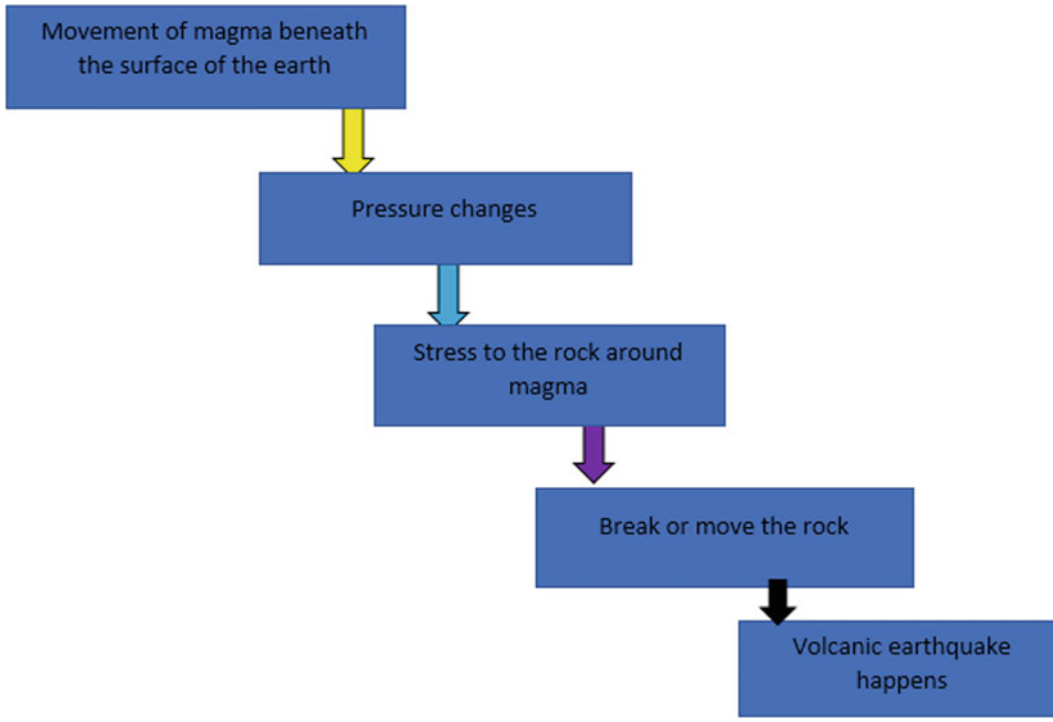


Fig. 3 A simplified waterfall diagram of how a volcanic earthquake happens

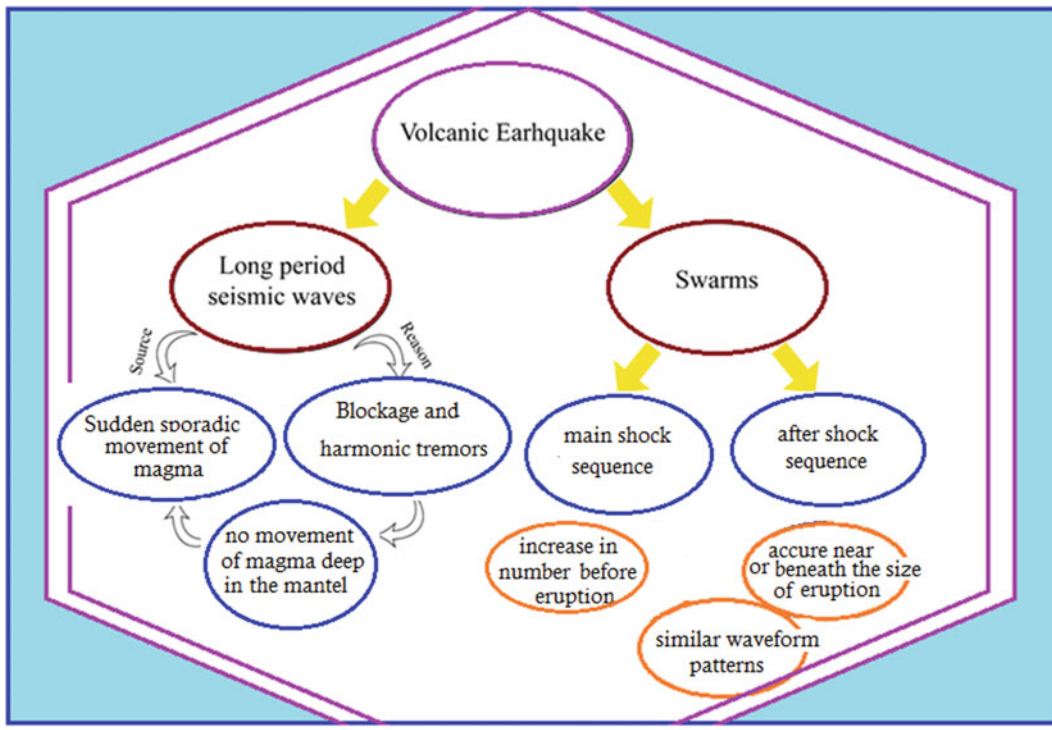


Fig. 4 Two main categories of volcanic earhquakes waves in a general schematic view

As the brains main structure is based on neurons and simultaneously it works in a fuzzy space, we can say it works by the fusion of Neural Networks and fuzzy logic which is known as "neuro-fuzzy system" in the literature.

The most brilliant power of natural neurons is their high ability to "**Learn**" from data. For volcanology and seismology we can take advantage of this power while also use fuzzy logic, because we have enough amount of data which an artificial neural network can learn from it. Also the available data is low prices and so fuzzy logic is a suitable tool for deal with this natural "vague" or "fuzzy" data. Consequently, **neuro-fuzzy** algorithms are suitable and useful tools to work on volcanological and seismological data (Hajian and Styles 2018).

1.1 Brief List of Intelligent Methods Applications in Volcanology and Seismology

1.1.1 In Volcanology

- Classification of volcano activity using Machine Learning (Hajian et al. 2019)
- Classification of volcanic deformation using INSAR data (Anantrasirichai et al. 2018)
- Volcano activity recognition using deep modular multi-dimodal fusion on multiple sensors (Le et al. 2018)
- Volcano hotspots detection using Adaptive Neurofuzzy Interference System (ANFIS) classifier (Rathnam and Ramashari 2016)
- Denoising of volcanological gravity and geo magnetic signals using ANFIS (Negro et al. 2008)
- Detecting volcano deformation from satellite imagery (Anantrasirichai et al. 2019)
- Fuzzification of Volcanic Explosivity index (VEI) using if–then fuzzy rules (Cagnoli 1998)
- Classification of isolated volcano-seismic events based on inductive transfer learning (Titos et al. 2020)
- Integrated inversion of ground deformation and magnetic data of Etna volcano using genetic algorithm (Currenti et al. 2007)
- Classification of state of volcanic activity using volcanic tremor data through Support Vector Machine using tremor data (Masotti et al. 2006)
- Classification of infrasonic events generated by active volcanoes (Cannata et al. 2011)
- Pattern recognition of volcanic tremor data using KK analysis (Messina and Langer 2011)
- Classification of volcanic ash particles using convolutional neural network and probability (Shoji et al. 2018)
- Automatic Classification of Volcano-Seismic Signatures using Machine Learning (Malfante et al. 2017)
- Shape recognition of volcanic ash by simple convolution neural network (Shoji and Noguchi 2017)
- Automatic recognition system for volcano-seismic events recognition through deep neural networks (Titos et al. 2018)
- Expert system for computer-guide volcano monitoring on Mt. Etna (Cannavo et al. 2017)

1.1.2 In Seismology

- Automatic classification of volcano-seismic signature using machine learning (Malfante et al. 2017)
- Seismic activity predication using computational intelligent techniques (Asim et al. 2017)
- Detecting "small" seismic events and addressing noisy data (Jiao and Alavi 2020, see Fig. 5a, b)
- Processing massive detected seismic data with sever noise to enhance the seismic performance of structures (Jiao and Alavi 2020)
- Intelligent recognition system for locating areas prone to strong earthquakes using fuzzy clustering zoning method (Gvishiani et al. 2016)
- Machine learning methods for seismic hazard forecast (Gitis and Derendyaev 2019)

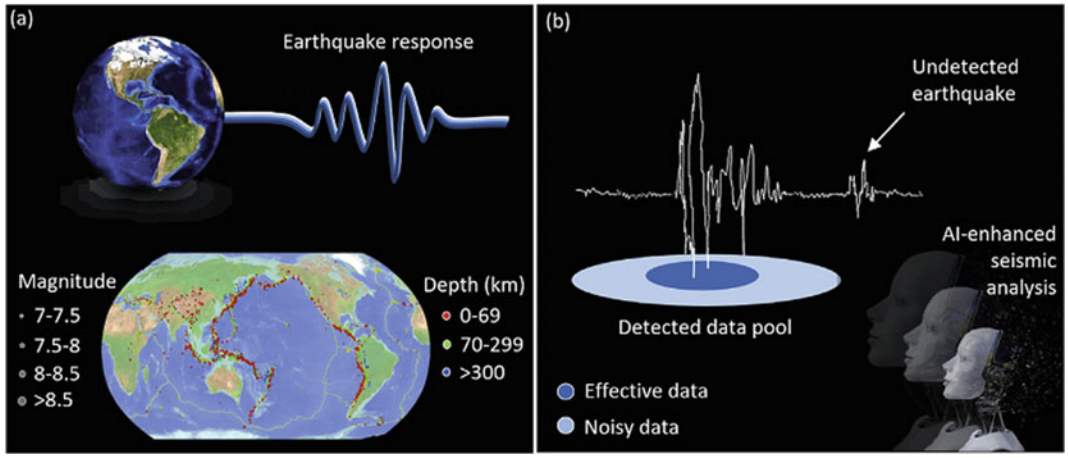


Fig. 5 **a** Characteristics of earthquake and seismic events occurred during 1900–2013 (USGS 2019), and **b** AI-enhanced seismic analysis in detecting "small" Seismic events and addressing noisy data (Jiao and Alavi 2020)

- Convolutional neural network for earthquake detection and location (Perol et al. 2018)
- High resolution imaging of fault zone and Deep Learning approach to seismic phase association (Ross et al. 2018)
- Compensating Absorption and Dispersion in Prestack Time Migration with Effective Q Estimation and Fresnel Zone Identification Based on Deep Learning (Wu et al. 2022)
- Extracting waveform features for similarity-based earthquake detection using Finger print And Similarity Thresholding (FAST); as a pattern mining approach (Bergen and Beroza 2019)
- Extracting dispersion curves from ambient noise correlation using deep learning (Zhang et al. 2020)
- Directivity modes of earthquake populations with Unsupervised Learning (Ross et al. 2020)
- Identifying the area of the focus of an expected earthquake using intelligent seismic acoustic system (Aliev 2017)
- Automatic micro-seismic event picking via unsupervised machine learning (Chen 2018).
- Earthquake rupture dynamics predication using machine learning (Ahmed and Daub 2019)
- Reliable real-time seismic signal/noise discrimination using Machine Learning (Meier et al. 2019)
- P-wave arrival picking and first motion polarity determination through deep learning (Ross et al. 2018)
- Unsupervised Feature Selection for pattern research in seismic time series using Self-Organizing Map (SOM) (Kohler et al. 2008)
- Seismic risk mitigation in Urban areas based on data mining techniques (Atanasiu 2008) (Fig. 6).

1.2 Metaheuristic Algorithms

In computer science and mathematical optimization, a metaheuristic is a higher-level procedure or heuristic designed to find, generate, or select a heuristic (partial search algorithm) that may provide a sufficiently good solution to an optimization problem, especially with incomplete or imperfect information or limited computation capacity (Balamurugan et al. 2015; Bianchi et al. 2009), Metaheuristics sample a set of solutions which is too large to be completely sampled.

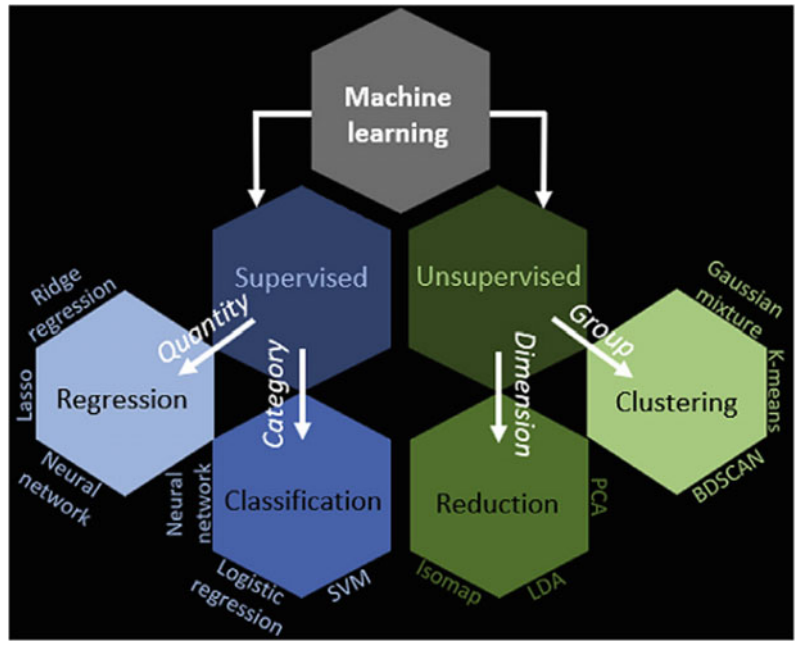


Fig. 6 Characteristics of machine learning enabled seismic model (Jiao and Alavi 2020)

Metaheuristics may make few assumptions about the optimization problem being solved, and so they may be usable for a variety of problems like the dynamic problems for natural complex phenomena as in seismology and volcanology. Compared to optimization algorithms and iterative methods, metaheuristics do not guarantee that a globally optimal solution can be found on some class of problems (Blum and Roli 2003). Many metaheuristics implement some form of stochastic optimization, so that the solution found is dependent on the set of random variables generated (Bianchi et al. 2009). There are a wide variety of metaheuristics, the Euler diagram of the different classifications of metaheuristics is depicted in Fig. 7. The Intelligent optimization based on Metaheuristic mostly used in Geophysics are sown in Fig. 8.

- Intelligent Optimization: { *Methahuristic* / *Evolutionary*
- Swarm intelligent PSO { *selforganization* / *informationflow*

1.3 Intelligent Methods

"The widespread use of intelligent method and its dramatic application in various scenes of everyday life indicates the human interest in inspiring divine intelligent creation, as if man sees the miracle of this creation in this new mirror much better than the old stained mirrors!" (Alireza Hajian).

When we say "Intelligent methods" we mean the intelligent techniques based on concepts of the artificial intelligent but not essentially artificial intelligence. Intelligent methods are data-based or data driven algorithms that explore data without any pre-assumption about the model of the phenomena. These techniques are mostly used when we have a suitable data bank of the experimental and/or simulated data of an unknown system whether natural or man-made. One of the important and powerful key tools to solve the problem of **nonlinear systems identification** is really the Intelligent Methods which have shown good results and implied their flexibility, adaptively and robust advantages.

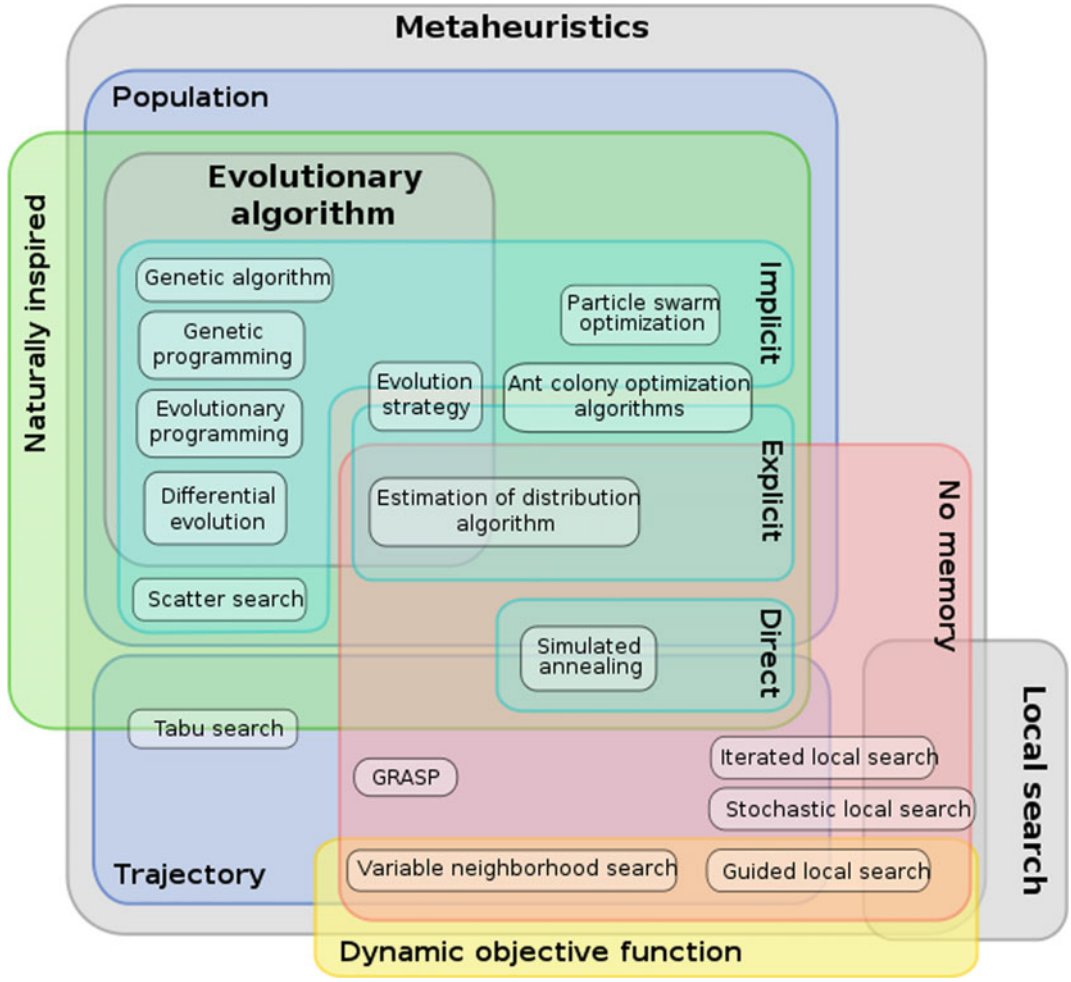


Fig. 7 Euler diagram of the different classifications of metaheuristic (after https://en.wikipedia.org/wiki/File:Metaheuristics_classification.svg)

Intelligence methods may be used for:

- Capturing individual and collective knowledge and extending a knowledge base, using artificial intelligence and database technologies
- Capturing tacit knowledge, using expert systems, case-based reasoning, and fuzzy logic
- Knowledge discovery, or discovering underlying, hidden patterns in data sets, using neural networks and data mining
- Generating solutions to highly complex problems, using genetic algorithms
- Automating routine tasks, using intelligent agents.

Artificial intelligent (AI) and ML can refine our understanding of deep earth's structures and seismic sources using a new glass from a new perspective view. In fact, this helps to improve earthquake detection, allowing for increased preparedness. Ross et al. (2018) used AI for earthquake monitoring, high-resolution imaging of fault zones, and exploring the physics of the earthquakes. Bergen and Beroza (2019) developed a new algorithm for automatically identifying weak earthquake signals in large seismic data sets. A Tree-Schematic of Intelligent methods applications in volcanology/seismology is shown in Fig. 9.

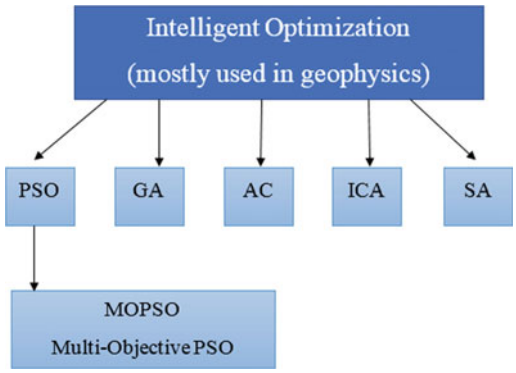


Fig. 8 Intelligent optimization based on Metaheuristic mostly used in Geophysics (where PSO stands for: Particle Swarm Optimization, GA: Genetic Algorithm, ICA: Imperialist Competitive Algorithm, SA: Simulated Annealing, AN: Ant Colony, SVM: Support Vector Machine)

1.3.1 Motivations

Generally, it is very much difficult to model complex volcano systems. This problem has some reasons: first our lack of "knowledge" not essentially "data", as i.e. we have lost of various data for some volcanos as Etna. But the classical methods are mostly based on pre-assumed model or are namely "model-based". In this type of methods, we assume a model with unknown parameters (i.e. coefficients) and try to fit the experimental data to the model to find the optimum model parameters. Here, the problem is that most of the pre-assumed models are so simple compared to the very complex system of volcano and/or earthquake.

Second reason is due to the fact that classical logic is too rigid or "crisp" while nature is "vague" or namely "fuzzy" (Cagnoli 1998), especially in geology there are lot of exceptions and intermediate characteristic (Cagnoli 1998). To solve this challenge one way is using "fuzzy logic" which is a branch of intelligent methods also known as "fuzzy systems". Human brain works as well using "fuzzy" logic framework. This means that human brain can work as well with a high accuracy and good precision using "uncertain" or "vague" input data (i.e. the human five senses which measure the environmental parameters "approximately"). The calculations and computing's based on "fuzzy logic" and/or "fuzzy arithmetic" is well-known as "soft computing" in the literature. A newer approach, of this intelligent type, is Computing With Words (CWW) in which all the computations are based on "words" not "numbers" even "fuzzy numbers" are not used, this recent concept was extended by Zadeh, God father of "fuzzy logic" and the founder of "fuzzy systems". The main advantage of CWW is no need to transform words into binary or digits and the processing and analyzing are both directly with words! In essence, **Computing with Words** (**CWW**) is a system of **computation** in which the objects of **computation** are predominantly **words**, phrases and propositions drawn from a natural language. **CWW** is based on fuzzy logic. In science there is a deep-seated tradition of according much more respect to numbers than to **words**.

Even with all that seismologists have learned up to now about the earthquake, new technologies show there are more interesting facts to be discovered specially from big data sets gathered for earthquakes all over the world via the **Global Seismographic Network (GSN)** (Fig. 10) using intelligent techniques.

Recent improvement in machine learning capabilities and the availability of large seismic data sets have opened new windows into the application of ML tools in seismological fields (Fig. 11) and also a wide reports contains data and information about the active volcanoes around the world namely "***Global Volcano Model Network (GVM)***". **GVM** is a growing international network that aims to create a sustainable, accessible information platform on volcanic hazard and risk. GVM will provide systematic evidence, data and analysis of volcanic hazards and risk on global and regional scales, and support Volcano Observatories at a local scale. GVM will develop capabilities to anticipate future volcanism and its consequences. "The GVM project will develop an integrated global database system on volcanic hazards, vulnerability and exposure, make this globally accessible and crucially involve the international volcanological community and users in a

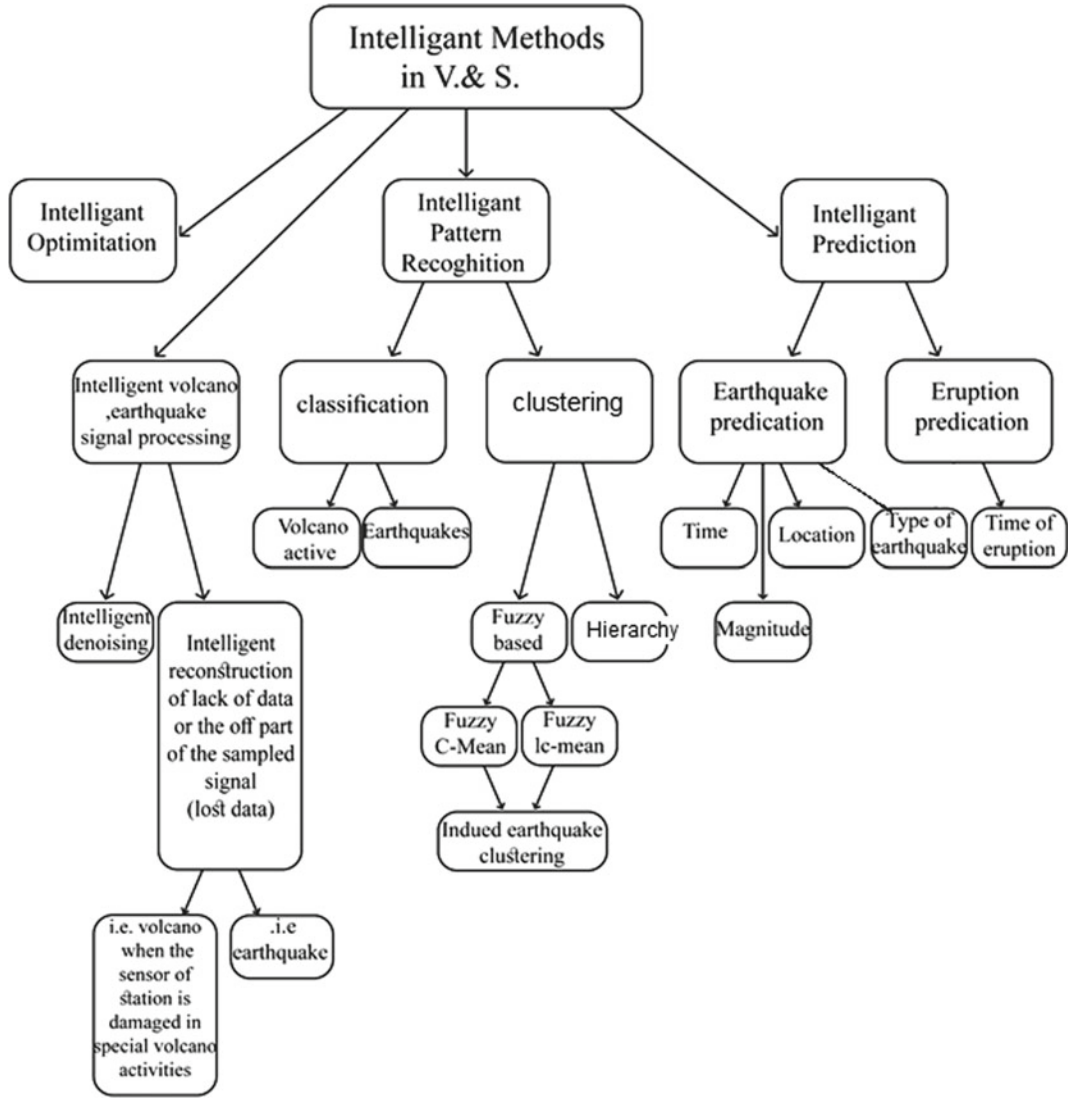


Fig. 9 A tree-schematic of intelligent methods applications in volcanology/seismology

partnership to design, develop analyses and maintain the database system. The GVM project will aim to establish new international **metadata** standards that will reduce ambiguity in the use of global volcanic datasets. Vulnerability and exposure data will be integrated into the GVM and again new methods of assessment and analysis will be investigated and tested. The project also intends to establish methodologies for analysis of the evidence and data to inform risk assessment, to develop complementary volcanic hazards models, and create relevant hazards and risk assessment tools (https://globalvolcanomodel.org/). Based on the powerful ability of the intelligent techniques to explore metadata Valade et al. (2019) proposed a plan for a Global Volcano Monitoring Using Multisensory Sentinel Missions and Artificial Intelligence, namely "MOUNTS" Monitoring System. They presented the volcano monitoring platform MOUNTS (Monitoring Unrest from Space), which aims for global monitoring, using multisensory satellite-

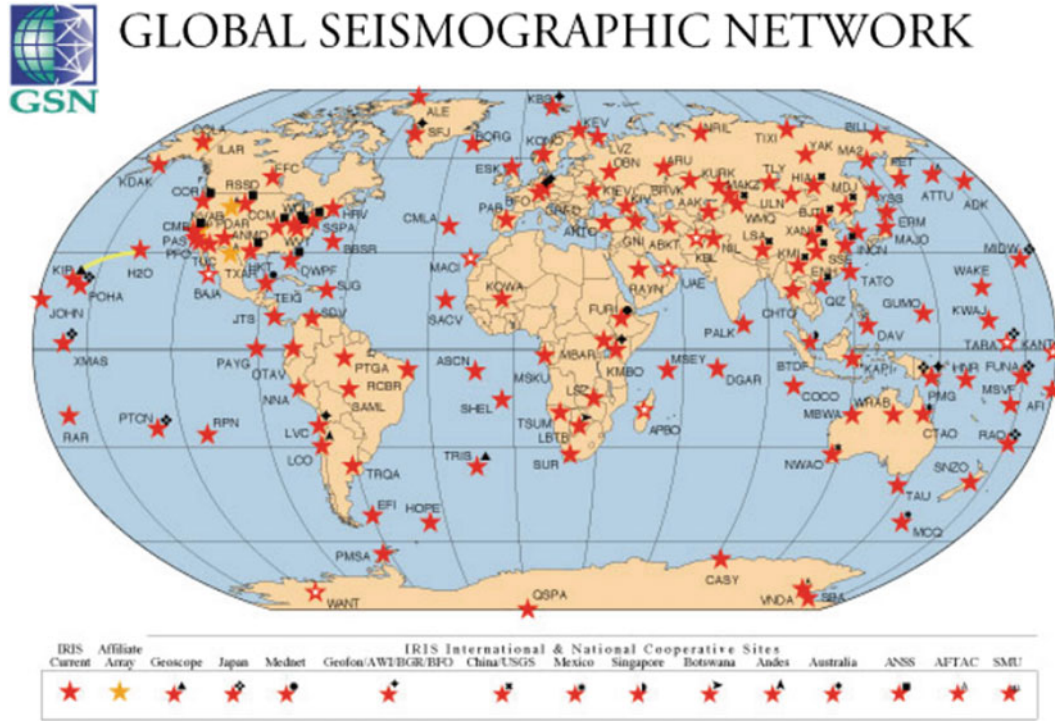


Fig. 10 Stations of the global seismographic network, including GSN-affiliated sites (figure courtesy of IRIS), see also Fig. 11

based imagery (Sentinel-1 Synthetic Aperture Radar SAR, Sentinel-2 Short-Wave InfraRed SWIR, Sentinel-5P TROPOMI), ground-based seismic data (GEOFON and USGS global earthquake catalogues), and artificial intelligence (AI) to assist monitoring tasks. It provides near-real-time access to surface deformation, heat anomalies, SO_2 gas emissions, and local seismicity at a number of volcanoes around the globe, providing support to both scientific and operational communities for volcanic risk assessment.

The research will provide the scientific basis for mitigation strategies, responses to ash in the atmosphere for the aviation industry, land-use planning, evacuation plans and management of volcanic emergencies.

Another important issue that motivates scientists to use intelligent methods in volcanology and seismology is the recent brilliant improvements in computer systems both in terms of hardware and software. Especially, the recent improvements in the huge capacity of virtual machines are helping the scientists to save and process a large amount of "**Big Data**" much faster than before. Therefore, the old problem of slow rate for running of deep layered neural networks with the aid of computers has been solved to some extent. Totally, in the light of dramatic improvement in computers speed of processing, the intelligent methods not essentially require much more time than that of classical methods.

There is a large variety of different behaviors of the volcanoes reference to the coupling of high nonlinearity and complexity of volcanic dynamic processes. This might lead to the same records, direct or indirect impacted by the volcano, for different states of the volcano activity that is known as non-uniqueness problem in geophysical "**inversion**" literature. Consequently, the fast volcano state assessment is sometimes impossible even for the expert personnel. In this way, intelligent methods can help to design an expert

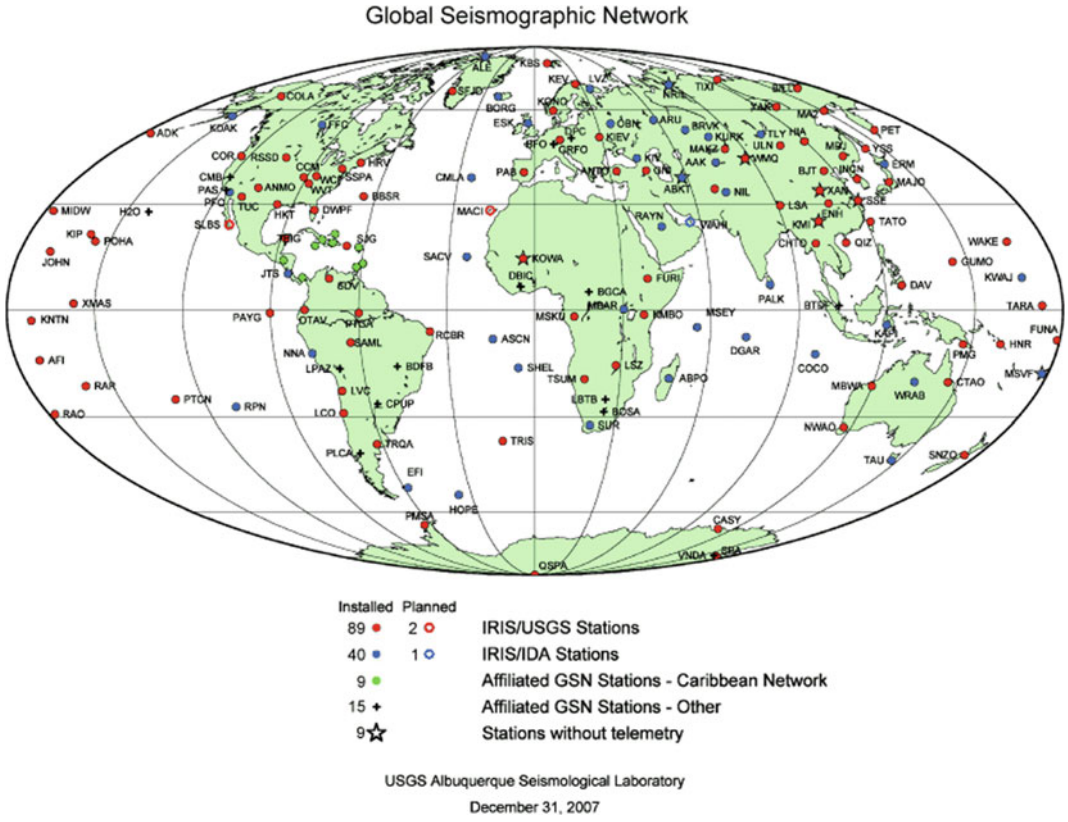


Fig. 11 Global seismographic network gathering a very huge volume of data all over the world, a very nice but so big data for exploring the earthquake phenomena using intelligent methods i.e. Machine Learning/Deep Learning (USGS 2007)

system based on probabilistic model which also can solve the problem of possible failures in recorded data. Furthermore, due to intrinsic uncertainties on volcanological data, intelligent methods help to make decision network to aid the interpreters achieving better interpretations of the volcano records, i.e. the on-line volcano records registered in the control /monitoring room, in this application it's very vital to do a fast interpretation of the online volcano records so if there is any hazard activity ahead to alert troops immediately (Cannova et al. 2017).

Fuzzy logic is a useful tool to model non-well-defined systems i.e. volcano and/or earthquake. On the other hand, it is possible to use data-based methods for complex systems like volcanos when there are enough various types of data gathered close to its craters (i.e. geophysical, geochemical, satellite images, etc.).

The schematic of a brief list of the various motivations to use intelligent methods in volcanology/seismology is shown in Fig. 12.

1.3.2 Machine Learning; an Overview

One of the most common classes of data-based methods is machine learning (**ML**). Supervised ML means that we design a machine which can learn from data (experimental and/or simulated). There are various ML's algorithms; generally, they are divided into two main groups: **Supervised** and **Unsupervised.** Machine Learning methods are categorized into two-main groups: **Supervised** and **Unsupervised**, the first category is used for "**classification**" and "**regression**" while the unsupervised is commonly

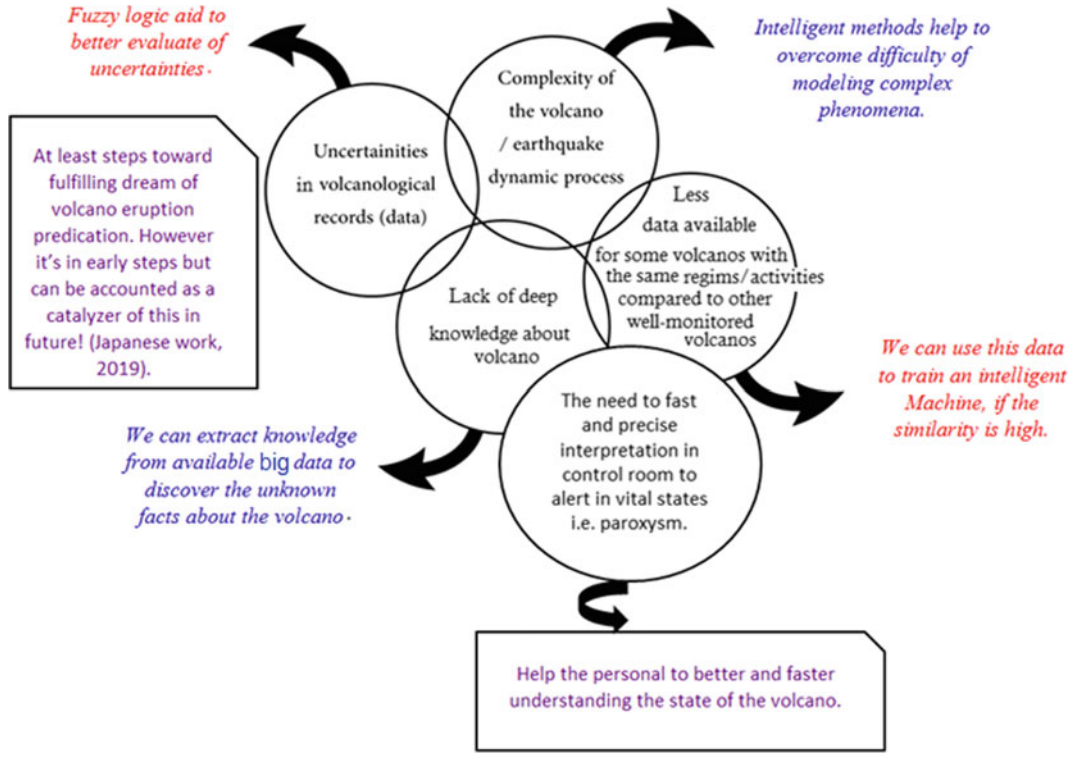


Fig. 12 A brief list of the various motivations to use intelligent methods in volcanology/seismology

used for "**clustering**" tasks. The classification of **ML** from this viewpoint is depicted in Fig. 13 with subdivisions of each category.

As we will describe the ML in Chapter "Machine Learning: The Concepts" in details, here briefly a simplified framework to machine learning, with **5 main areas of the machine learning process** is listed below (ref: https://www.kdnuggets.com/2018/05/general-approaches-machine-learning-process.html):

Step1—Data collection and preparation: everything from choosing where to get the data, up to the point it is clean and ready for feature selection/engineering.

Step2—Feature selection and feature engineering: this includes all changes to the data from once it has been cleaned up to when it is ingested into the machine learning model.

Step3—Choosing the machine learning algorithm and training our first model: getting a "better than baseline" result upon which we can (hopefully) improve.

Step4—Evaluating our model: this includes the selection of the measure as well as the actual evaluation; seemingly a smaller step than others, but important to our end result.

Step5—Model tweaking, regularization, and hyper parameter tuning: this is where we iteratively go from a "good enough" model to our best effort.

Based on these global stages for machine learning process Jaio and Alavi (2020) presented main calculation steps of machine learning enabled seismic model which is presented in Fig. 14.

In the next four chapters of the book the recent hot trends of Neural and fuzzy based systems, Machine learning, deep learning and metaheuristic algorithms will be presented with interesting modern applications in both volcanology and seismology and step-by-step guidance of their design and test in Matlab software.

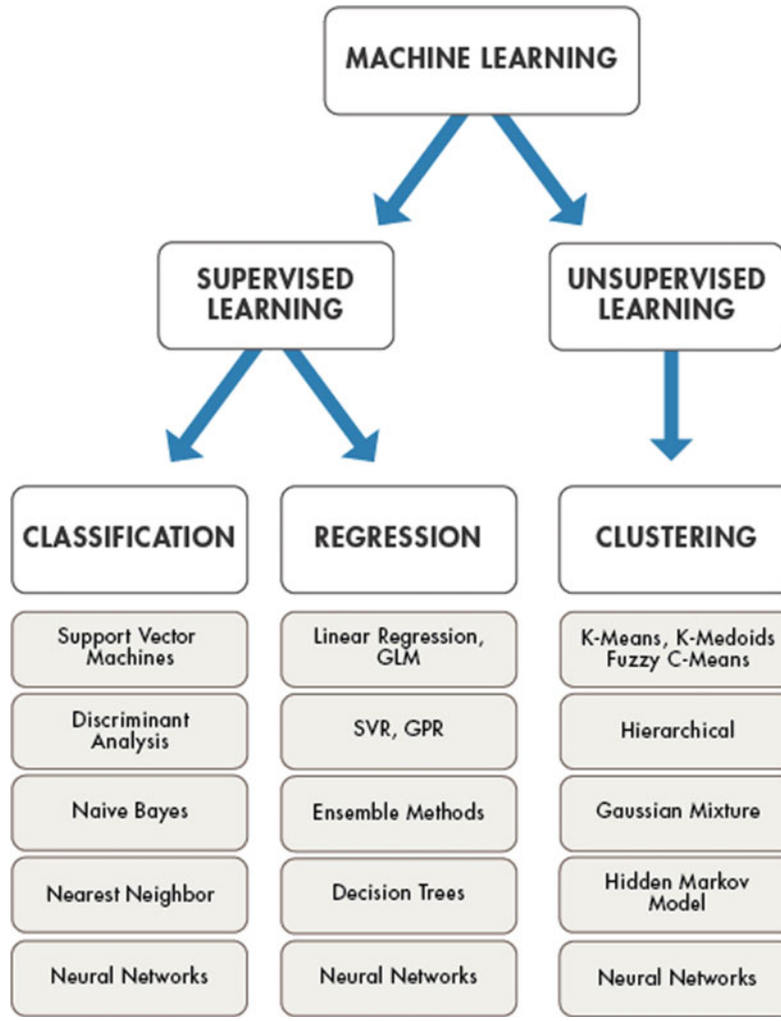


Fig. 13 Classification of machine learning methods

1.4 The Role of Intelligent Methods Toward Big Volcano Science

Papale and Grag (2022) contend that a comparative "**Big Science"** approach will progressively concern volcano science, and briefly portray three cases of advancements in volcanology requiring such an approach will characterize the current decade (2020–2030): the Krafla Magma Testbed activity; the advancement of a Global Volcano Science Simulator; and the developing pertinence of "Big Science" in volcano science. With the advancement of the computerized age, big information and related innovations such as Machine Learning (ML) and artificial intelligence (AI) have exploded in essentially any angle of science (e.g., Chen et al. 2012; Wamba et al. 2015; Gorelick et al. 2017). AI algorithms can be prepared to replicate a few of our capabilities i.e. driverless cars, online exchange of an important talk into text, etc. What looks more important in volcano science, in any case, is that ML and AI calculations can be utilized to discover, hidden within enormous groupings of information, important designs that trained groups of people may miss in months or a long time of work. A sample of KMT concept

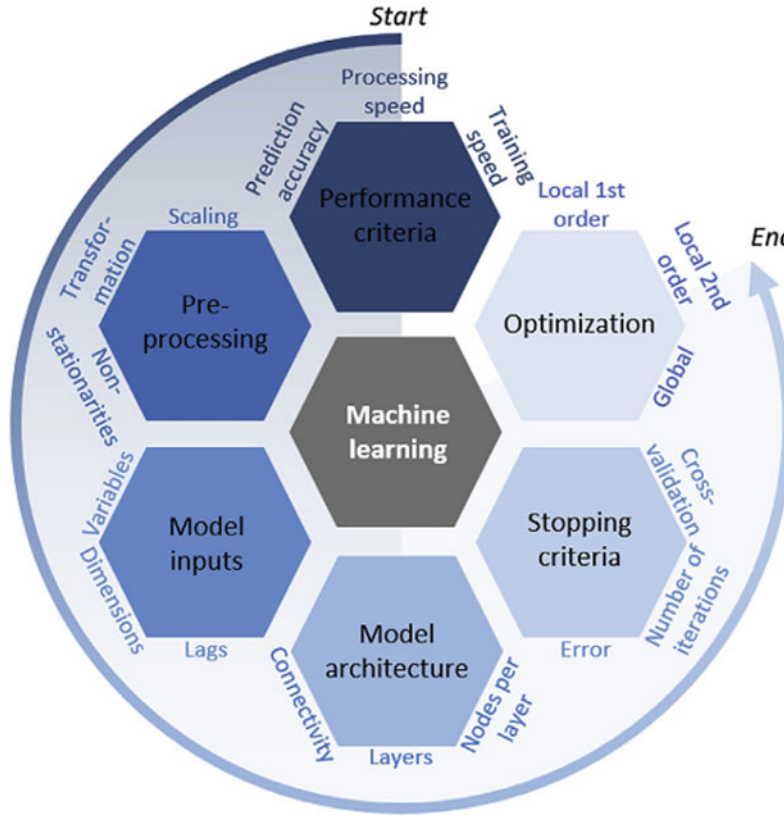


Fig. 14 Main calculation steps of machine learning enabled seismic model (Jiao and Alavi 2020)

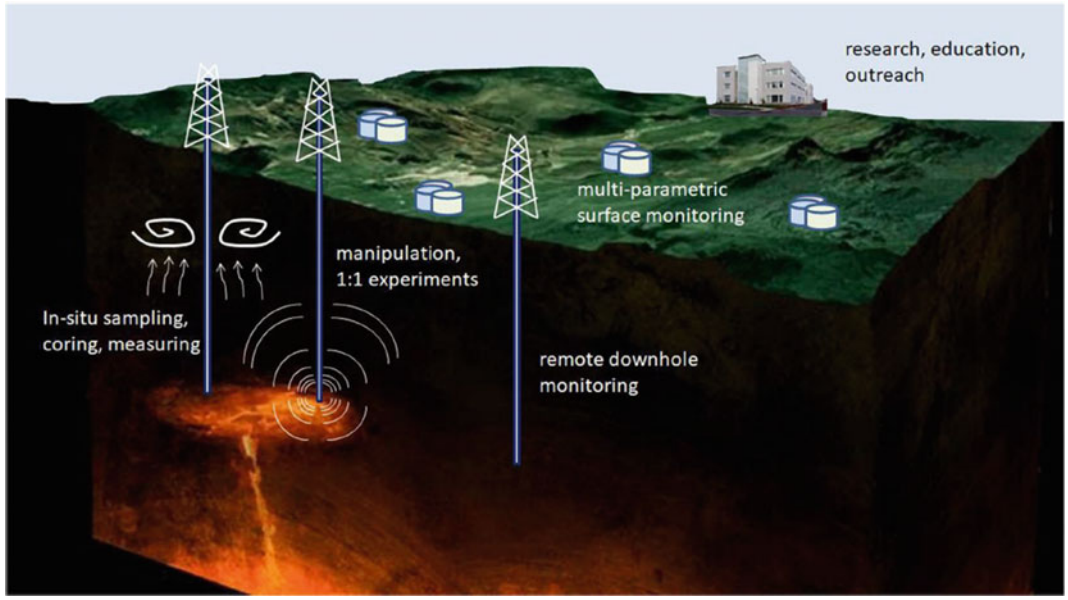


Fig. 15 The KMT concept. A series of wells are kept open inside and around the shallow magma intrusion at Krafla (2.1 km depth). Temperature-and corrosion-resistant instrumentation is placed inside the wells down to magma. The surface is heavily instrumented with an advanced multi-parametric monitoring network. Dedicated laboratories, offices, and a visitor center complement the infrastructure. Background picture: courtesy of GEORG (Geothermal Research Cluster of Iceland) (Papale and Graj 2022)

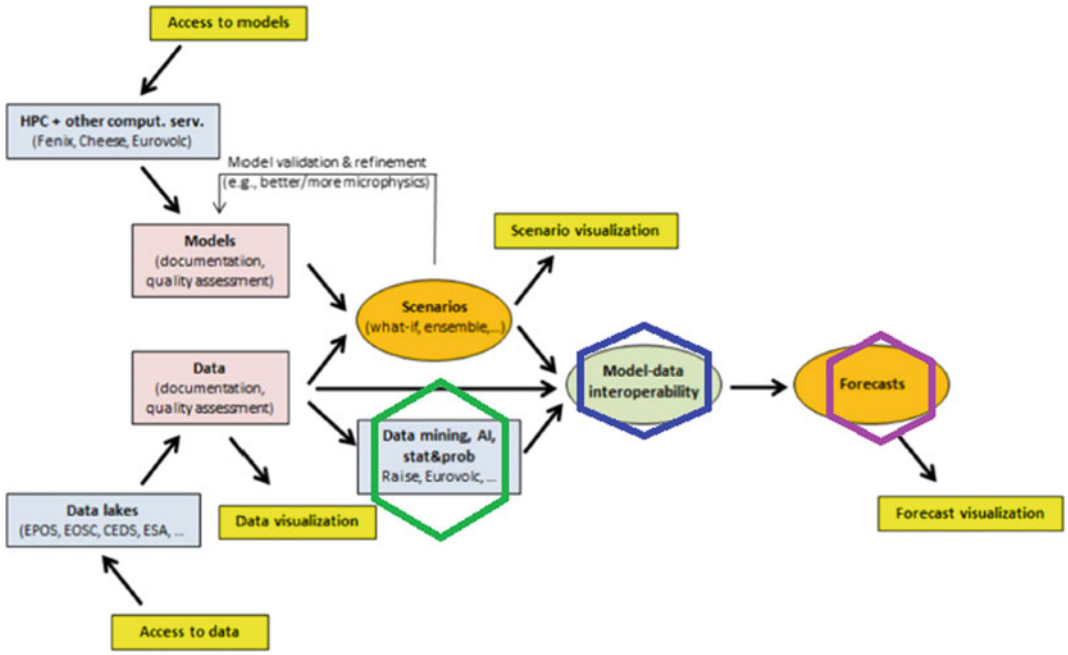


Fig. 16 Possible scheme for a digital twin of a volcanic system finally aims to forecast volcano behavior and activities, proposed by Papale and Grag (2022), the parts that Intelligent Methods plays important roles are overmarked by colored hexagonal, AI can extract hidden knowledge from big Data and so plays a brilliant role to improve "Big Science" of volcanology Papale and Grag (2022)

contains a series of wells is shown in Fig. 15. AI can extract hidden knowledge from big Data and so plays a brilliant role to improve "Big Science" of volcanology. A possible scheme for a digital twin of a **volcanic system** which finally aims to forecast volcano behavior and activities is depicted in Fig. 16.

References

Ahamed S, Daub EG (2019) Machine learning approach to earthquake rupture dynamics. ArXiv. https://doi.org/10.48550/arXiv.1906.06250

Aliev T (2017) Intelligent seismic-acoustic system for identifying the area of the focus of an expected Earthquake. In: Earthquakes—tectonics, hazard and risk mitigation. London, United Kingdom: IntechOpen. Available: https://www.intechopen.com/chapters/52427. https://doi.org/10.5772/65403

Anantrasirichai N, Biggs J, Albino F, Hill P, Bull DR (2018) Application of machine learning to classification of volcanic deformation in routinely generated InSAR data. J Geophys Res Solid Earth 123:6592–6606. https://doi.org/10.1029/2018JB015911

Anantrasirichaia N, Biggsb J, Albinob F, Bulla D (2019) A deep learning approach to detecting volcano deformation from satellite imagery using synthetic datasets. Remote Sens Environ. 111179. https://doi.org/10.1016/j.rse.2019.04.032

Asim KM, Awais M, Martínez-Álvarez F, Iqbal T (2017) Seismic activity prediction using computational intelligence techniques in northern Pakistan. Acta Geophys 65:919–930. https://doi.org/10.1007/s11600-017-0082-1

Atanasiu GM (2008) Seismic risk mitigation in urban areas based on artificial intelligence methods. In: Proceedings of the 14th world conference on earthquake engineering. http://invenio.itam.cas.cz/record/9443/files/09-01-0057.pdf

Balamurugan R, Natarajan AM, Premalatha K (2015) Stellar-mass black hole optimization for biclustering microarray gene expression data. Appl Artif Intell 29 (4):353–381. https://doi.org/10.1080/08839514.2015.1016391

Bergen KJ, Beroza GC (2019) Earthquake fingerprints: extracting waveform features for similarity-based earthquake detection. Pure Appl Geophys 176:1037–1059. https://doi.org/10.1007/s00024-018-1995-6

Bianchi L, Dorigo M, Gambardella LM, Gutjahr WJ (2009) A survey on metaheuristics for stochastic combinatorial optimization. Nat Comput 8(2):239–287. https://doi.org/10.1007/s11047-008-9098-4

Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput Surv 35(3):268–308

Cagnoli B (1998) Fuzzy logic in volcanology. Episodes 19(2):94–96. https://doi.org/10.18814/epiiugs/1998/v21i2/004

Cannata A, Montalto P, Aliotta M, Cassisi C, Pulvirenti A, Privitera E, Patanè D (2011) Clustering and classification of infrasonic events at Mount Etna using pattern recognition techniques. Geophys J Int 185(1):253–264. https://doi.org/10.1111/j.1365-246X.2011.04951.x

Cannavo F, Cannata A, Cassisi C, Di Grazia G, Montalto P, Prestifilippo M, Eugenio P, Gambino S, Coltelli M (2017) 19th EGU general assembly, EGU2017. In: Proceedings of the conference. Vienna, Austria., p 7486

Chen H, Chiang RHL, Storey VC (2012) Business intelligence and analytics: from big data to big impact. MIS Q 36(4):1165–1188

Chen Y. (2018) Automatic microseismic event picking via unsupervised machine learning, Geophys J Int 212 (1):88–102. https://doi.org/10.1093/gji/ggx420

Currenti G, Del Negro C, Fortuna L, Ganci G (2007) Integrated inversion of ground deformation and magnetic data at Etna volcano using a genetic algorithm technique. Ann Geophys 50(1):21–30, 25 Dec 2007. Available from: https://www.annalsofgeophysics.eu/index.php/annals/article/view/3082

Gitis VG, Derendyaev AB (2019) Machine learning methods for seismic hazards forecast. Geosciences 9 (7):308. https://doi.org/10.3390/geosciences9070308

Gorelick N, Hancher M, Dixon M, Ilyushchenko S, Thau D, Moore R (2017) Google Earth engine: planetary-scale geospatial analysis for everyone. Remote Sens Env 202:18–27

Gvishiani AD, Dzeboev BA, Agayan SM (2016) FCaZm intelligent recognition system for locating areas prone to strong earthquakes in the Andean and Caucasian mountain belts. Izv Phys Solid Earth 52:461–491. https://doi.org/10.1134/S1069351316040017

Hajian A, Styles P (2018) Application of soft computing and intelligent methods in geophysics. Springer International Publishing AG, part of Springer Nature. https://doi.org/10.1007/978-3-319-66532-0

Hajian A, Cannavò F, Greco F, Nunnari G (2019) Classification of Mt Etna (Italy) volcanic activity by machine learning approaches. Ann Geophys 62(2): VO231

https://en.wikipedia.org/wiki/File:Metaheuristics_classification.svg

https://globalvolcanomodel.org/

https://www.annalsofgeophysics.eu/index.php/annals/article/view/8049

https://www.seismosoc.org/wp-content/uploads/2019/01/Machine-Learning-Flyer-v2.pdf

Jiao P, Alavi AH (2020) Artificial intelligence in seismology: Advent, performance and future trends. Geosci Front 11(3):739–744. https://doi.org/10.1016/j.gsf.2019.10.004

Kohler A, Ohrnberger M, Riggelsen C, Scherbaum F (2008) Unsupervised feature selection for pattern search in seismic time series, JMLR: workshop and conference proceedings. New Challenges Feature Sel 4:106–121

Le HV, Murata T, Iguchi M (2018) Volcano activity recognition using deep modular multi-dimodal fusion on multiple sensors, machine learning and knowledge discovery in databases. In: European conference, ECML PKDD 2018, Dublin, Ireland, 10–14 Sept 2018, Proceedings, Part III

Malfante M, Dalla Mura M, Mars J, Metaxian JP (2017) Machine learning for automatic classification of volcano-seismic signatures. In: 25th European signal processing conference, pp 2457–2461

Masotti M, Falsaperla S, Langer H. Spampinato S, Campanini R (2006) Application of support vector machine to the classification of volcanic tremor at Etna, Italy. Geophys Res Lett 33(20), CiteID L20304. https://doi.org/10.1029/2006GL027441

Meier M-A, Ross ZE, Ramachandran A, Balakrishna A, Nair S, Kundzicz P, Li Z, Andrews J, Hauksson E, Yue Y (2019) Reliable real-time seismic signal/noise discrimination with machine learning. J Geophys Res Solid Earth 124:788–800. https://doi.org/10.1029/2018JB016661

Messina A, Langer H (2011) Pattern recognition of volcanic tremor data on Mt. Etna (Italy) with KK Analysis—A software program for unsupervised classification. Comput Geosci 37(7):953–961. https://doi.org/10.1016/j.cageo.2011.03.015

Del Negro C, Greco F, Napoli R, Nunnari G (2008) Nonlinear processes geophysics 15:735–749. www.nonlin-processes-geophys.net/15/735/2008/

Papale P, Garg D (2022) Big volcano science: needs and perspectives. Bull Volcanol 84(20):1–7. https://doi.org/10.1007/s00445-022-01524-0

Perol T, Gharbi M, Denolle M (2018) Convolutional neural network for earthquake detection and location. Sci Adv 4(2):e170057. https://doi.org/10.1126/sciadv.1700578

Rathnam SM, Ramashri T (2016) Identification of volcano hotspots in multi spectral ASTER satellite images using DTCWT image fusion and ANFIS classifier. Am J Eng Res (AJER) 5(12):21–31

Ross ZE, Meier M-A, Hauksson E (2018) P wave arrival picking and first-motion polarity determination with deep learning. J Geophys Res Solid Earth 123:5120–5129. https://doi.org/10.1029/2017JB015251

Ross ZE, Trugman DT, Azizzadenesheli K, Anandkumar A (2020) Directivity modes of Earthquake populations with unsupervised learning. J Geophys Res Solid Earth 125(2). Art. No. e2019JB018299. ISSN 2169-9313. https://doi.org/10.1029/2019JB018299

Shoji D, Noguchi R, Otsuki S (2018) Classification of volcanic ash particles using a convolutional neural network and probability. Sci Rep 8:8111. https://doi.org/10.1038/s41598-018-26200-2

Shoji D, Noguchi R (2017) Shape recognition of volcanic ash by simple convolutional neural network. ArXiv, abs/1706.07178

Titos M, Bueno A, García L, Benítez C (2018) A deep neural networks approach to automatic recognition systems for volcano-seismic events. IEEE J Sel Topics Appl Earth Observations Remote Sens 11(5):1533–1544. https://doi.org/10.1109/JSTARS.2018.2803198

Titos M, Bueno A, García L, Benítez C, Segura JC (2020) Classification of isolated volcano-seismic events based on inductive transfer learning. IEEE Geosci Remote Sens Lett 17(5):869–873. https://doi.org/10.1109/LGRS.2019.2931063

USGS (2007) https://www.usgs.gov/programs/earthquake-hazards/gsn-global-seismographic-network

USGS (2019) Seismicity of the earth maps 1900–2013. US Geological Survey. Available at https://earthquake.usgs.gov/earthquakes/byregion/

Valade S, Ley A, Massimetti F, D'Hondt O, Laiolo M, Coppola D, Loibl D, Hellwich O, Walter TR (2019) Towards global volcano monitoring using multisensor sentinel missions and artificial intelligence: the MOUNTS monitoring system. Remote Sens 11:1528. https://doi.org/10.3390/rs11131528

Wamba SF, Akter S, Edwards A, Chopin G, Gnanzou D (2015) How big data can make big impact: findings from a systematic review and a longitudinal case study. Int J Prod Econ 165:234–246

Wu J, Shi Y, Guo A, Lu P, Yang Q (2022) Compensating absorption and dispersion in prestack time migration with effective Q estimation and fresnel zone identification based on deep learning. Front Earth Sci 18 Jan 2022. https://doi.org/10.3389/feart.2021.771570

Zhang X, Jia Z, Ross ZE, Clayton RW (2020) Extracting dispersion curves from ambient noise correlations using deep learning. IEEE Trans Geosci Remote Sens 58 (12):8932–8939. https://doi.org/10.1109/TGRS.2020.2992043

Machine Learning: The Concepts

Abstracts

The purpose of this chapter is to provide the description of some of the main algorithms used in the field of Machine Learning. The mathematical details are reduced to the minimum necessary in order to give the readers a clear idea of the key points and allow them to proceed in their applications with greater awareness.

1 Introduction

The purpose of this chapter is to provide the necessary knowledge on Machine Learning (ML). As the reader will understand reading this chapter, ML essentially consists of a set of tools that are applied for data processing, with the aim of establishing cause-and-effect connections or looking for hidden structures in the data itself. As the number of ML algorithms has rapidly grown since the 1950s, even giving a brief description is not an easy task in the limited extension of a book chapter. Fortunately, very good and comprehensive books there exist such as (Bishop 2006; Hastie et al. 2008; Goodfellow et al. 2016; Lei et al. 2017), to which the readers are addressed for more deeply understanding. Considering the books cited above and many others that would take a long time to cite, the readers of this chapter may reasonably wonder what the purpose of this chapter is. The answer is simple. This chapter only wants to summarize, avoiding many mathematical details, the rationale of the main methods of ML. This will facilitate the understanding of the applications presented in the Chapter "Machine Learning Applications in Volcanology and Seismology" of the book and of course the vast literature already existing. A feature shared by ML algorithms is to learn from data, using techniques that in the vast majority make use of statistics. Broadly speaking, ML techniques can be grouped into two categories, referred to as supervised and unsupervised, depending on whether the data used is labeled or not, as schematically shown in Fig. 1.

In turn, supervised techniques are divided into classification and regression algorithms, according to whether the target is categorical or real valued. Similarly, unsupervised algorithms are divided into clustering or dimensionality reduction algorithms, depending on whether the goal is to group the data into sets of similar elements or reduce the size of the data set. Many techniques used for classification are also used for regression with some tricks. Popular algorithms for classification are Bayesian Classifiers, which can be distinguished in several variants (e.g. Linear and Quadratic classifiers, Naive Bayes classifiers, etc.), K-Nearest Neighbor classifiers (KNN), Parzen density classifiers, Decision Tree (DT), Artificial Neural Networks (ANN), Support Vector Machine (SVM), Random Forest (RF), just to mention a few. Hidden Markov Models (HMM) are special classifiers that can be trained

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

A. Hajian et al., *Intelligent Methods with Applications in Volcanology and Seismology*, Advances in Volcanology, https://doi.org/10.1007/978-3-031-15432-4_2

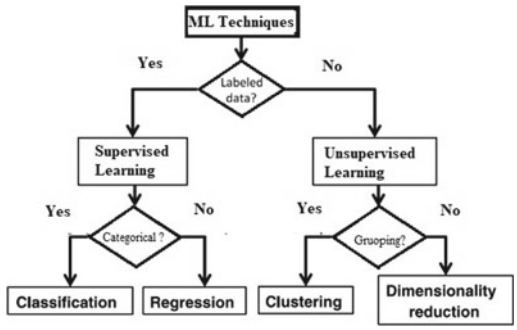


Fig. 1 A schematic grouping of ML techniques (Kong et al. 2019)

to assign a sequence of labels given a sequence of observations.

As concerning clustering techniques, popular approaches include Hierarchical clustering, K-means, Fuzzy C-means, Gaussian Mixture Models (GMM), Self-Organizing Maps (SOM). In particular, ANNs are available in several structures ranging from the simple Multi-Layer Perceptrons (MLPs) to more complex ones, such as Deep Neural Networks (DNN). Among DNN, the Convolution Neural Networks (CNN) have drawn considerable attention for applications. Reference to the fact that deep learning is an international trend in literature, we have separately explained both the concepts and applications of deep learning in volcanology and seismology in Chapters "Deep Learning: The Concepts" and "Deep Learning: Applications in Seismology and Volcanology".

Some most popular techniques for dimensionality reduction of dataset are the Principal Component Analysis (PCA), the Self-Organizing Maps (SOMs), the Independent Component Analysis (ICA). A short description of some of these techniques will be provided in the next sections.

Developers and/or scientists can count on a huge amount of tools, mostly free available, that can help them to build their own applications. However, it is very difficult to consciously apply these methods without having a sufficient understanding of their features and limits. Therefore, one of the purposes of this chapter is to fill this gap.

The process of building a model using ML algorithms always starts from the assumption that a dataset, sufficiently representative of the process under modeling is available. In classification and regression problems, the objects inside the dataset are usually referred to as *examples*, in order to indicate that they consist of input–output data associations. The input represents a set of values and/or classes to which we want to match other values or classes. For example, a seismologist who has collected a set of seismic waveforms may wish to know if it is tectonic or it should be attributed to a volcanic process or, again it is due to anthropic activities (e.g. an explosion). In this example, the input data will be real values that express some features of the wave-form while the output data will be classes (i.e. the categories to which the waveforms belong).

In clustering problems, the output data is not necessary, as in this case the problem is to find out if a data set has internal structures, that is, for example, if they can be grouped into two or more classes. In this case, referring again to the example of seismic waveforms, a seismologist who has recorded a not labeled dataset, might be interested to know in how many homogeneous groups they can be separated. A preliminary estimation of the best number of clusters that can be separated into a given dataset can be obtained by applying one of several criteria such as Davies-Bouldin criterion (Davies and Bouldin 1979) or the Silhouette analysis (Rousseeuw 1987).

In the case of classification and/or regression problems it is essential to divide the available data set into at least two sets which are respectively referred to as the *training* set and the *test* set. The training set will be considered in order to build a model whose performance will be assessed referring to examples contained into the testing set. If the model is considered reliable enough, then it can be used to process a new data set. However, since building a model there is a natural tendency to make choices which involve not only the training set but also the test set, it is a good practice to have a third set of data, typically referred to as the validation set, which allows to make the performance evaluation more

objective. Performance evaluation is a crucial point in ML applications. The reader must in fact always bear in mind that an ML algorithm always produces a model as a result, but is not normally able on its own to guarantee its reliability! It is therefore necessary to resort to the use of appropriate performance indices to objectively evaluate the reliability of the model. Some of these indices are described in Sect. 5.

The chapter is organized in this way. In Sect. 2 we give an overview of the state estimation problem. In Sect. 3 we address the problem of estimating the probability densities by using both parametric and non-parametric approaches. In Sect. 4 we describe relevant approaches to perform the supervised classification. In Sect. 5 we deal with the problem of evaluating the performance of supervised classifiers. In Sect. 6 we give an overview of the unsupervised classification. In Sect. 7 we describe the use of the Principal Component Analysis and the Self-Organizing Maps to reduce the dimensionality of a dataset. Finally, in Sect. 8 we mention some useful software tools that have been used through the preparation of this chapter.

2 An Overview of the State Estimation Problem

In many application fields, seismology and volcanology are among them, the process to infer some properties of a physical system, can be represented as in Fig. 2.

As described in (Lei et al. 2017), who we referred to for the description of this problem, it is postulated that the system is characterized by an unknown state x, that assumes values in some domain. In order to estimate x, appropriate measurements are performed, which produces vectors, here indicated as z, usually real-valued, i.e. $z \in R^M$. The term regression is used when the task is that of approximating a map f: $z \in R^N \rightarrow x \in R^K$, while the term classification is used when x is a discrete finite set, i.e. $x \in \Omega = [\omega_1, ..., \omega_K]$. Regression and classification are similar processes because they both aim to obtain an estimation $\hat{x}$ of the system state.

In several cases, probability theory is a solid ground to solve the above-mentioned estimation problem. In this framework the unknown state x is considered a random vector and $p(x)$ is the associated prior probability density. Generically speaking, the goal is to estimate the conditional probability density $p(x|z)$ which expresses the knowledge that we have on x after having observed z. In many cases, this is done through the following steps:

1. The prior probability density on x is first estimated by using the available training set.
2. The overall probability density of z is found by averaging the conditional density over the complete state space:

$$p(z) = \int_x p(z|x)p(x)dx \tag{1}$$

3. The Bayes theorem for conditional probabilities gives the posterior probability density $p(x|z)$:

$$p(x|z) = \frac{p(z|x)p(x)}{p(z)} \tag{2}$$

The steps above simplify in case of classification problems. Indeed in this case, being $x \in \Omega = [\omega_1, ..., \omega_K]$, the prior density $p(x)$ estimation (step 1) is equivalent to estimate the prior probabilities $P(\omega_k)$, $k \in 1, ..., K$, which is quite

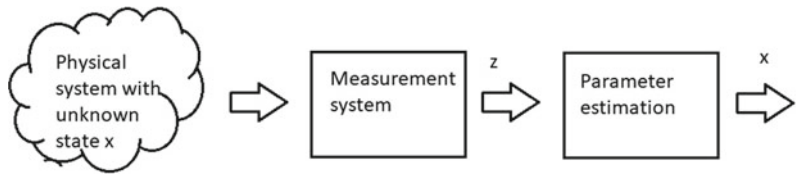


Fig. 2 The parameter estimation scheme

simple given a labeled training data set. Estimation of $p(\mathbf{z})$, assuming the classes mutually exclusive, can be written as:

$$p(z) = \sum_{k=1}^{K} p(z|\omega_k)P(\omega_k) \tag{3}$$

and finally step 3 can be rewritten as:

$$P(\omega_k|z) = \frac{p(z|\omega_k)P(\omega_k)}{p(z)} \tag{4}$$

This explains how it is possible to associate a probability that $x = \omega_k$ after observing z. It is trivial to understand that the key point to apply a probabilistic approach is that of estimating the conditional probability densities $p(z|\omega_k)$. This is explained with more details in the next section.

3 Parametric and Non-parametric Estimation of Densities

As mentioned in the previous section, in order to estimate the posterior probability density $p(x|z)$ we need to know the conditional probability density $p(z|x)$ and the prior probability $p(x)$. To simplify the notation, we consider the case of classification, for which we can write $p(z|x)$ as $p(z|\omega_k)$. To compute probability densities, both parametric and non-parametric approaches are available.

3.1 A Parametric Approach

In the parametric approach it is assumed that the kind of data distribution is known while the parameters of the probability density function are unknown. For example, assuming that data are *normally* distributed, the only unknown parameters are those of a Gaussian function, namely the mean vector μ and the covariance matrix C.

In order to estimate $p(z|\omega_k)$, $k = 1,...K$, let us suppose that the training set T_s is split according to their true class, as indicate in expression (5):

$$T_k = \{z_n, \theta_n | \theta_n = \omega_k\}, k = 1, \cdots N_k \tag{5}$$

where for each sample z_n, $\theta_n \in \Omega$, indicates the true class. Of course, if we indicate as N_s the total number of samples in the training set T_s and as N_k the number of samples in T_k, the following expression must hold:

$$N_s = \sum_{k=1}^{K} N_k \tag{6}$$

Indicating as z_1, z_2,. ... z_{Nk} the set of measures, assumed to be statistically independent, that can be associated with class ω_k and as α_k the parameters vector of the probability density associated to the sub-training set T_k, the following relation must hold:

$$p(z_1, z_2, \ldots, z_{N_k}|\omega_k, \alpha_k) = \prod_{n=1}^{N_k} p(z_n|\omega_k, \alpha_k) \tag{7}$$

Expression (7) says that the original problem is now that of computing the parameter α_k of the conditional probability $p(z_n|\omega_k)$.

This task is particularly simple if one assumes that data follows a known probability distribution density. For instance, in case of a gaussian distribution density, indicating as μ_k and C_k the unknown expectation vector and covariance matrix, it is possible to write:

$$p(z_n|\omega_k, \mu_k, C_k) = \frac{1}{\sqrt{(2\pi)^N |C_k|}} \exp\left\{-\frac{1}{2}(z_n - \mu_k)^T C_k^{-1}(z_n - \mu_k)\right\} \tag{8}$$

which can be easily evaluated once μ_k and C_k are known. This last task can be easily performed considering that unbiased estimation of μ_k and C_k can be obtained by using the following expressions:

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N_k} z_n \tag{9}$$

$$\hat{C}_k = \frac{1}{N_k - 1} \sum_{n=1}^{N_k} (z_n - \mu_k)(z_n - \mu_k)^T \quad (10)$$

From the user's point of view, it may be useful to observe that despite the simplicity of the expressions (9) and (10), their estimation requires an adequate number of samples. This is also a key point in order to avoid problems with computation of C_k^{-1}. As a rule of thumb, a commonly accepted rule is that the number of samples in each class must be $N_k > 5\ N,\ k = 1, ...,K,\ N$ being the z dataset dimension.

Another useful aspect is that in order to reduce the sensitivity to statistical errors in the calculation of C_k^{-1}, it is good practice to carry out the so-called regularization, which consists in applying one of several expressions similar to the following one:

$$C_k^{-1} = V_k((1 - \gamma)\Lambda + \gamma \frac{trace(\Lambda)}{N} I)^{-1} V_k^T, 0 \le \gamma \le 1 \quad (11)$$

where $\frac{trace(\Lambda)}{N}$ is the average of the eigenvalues of $\widehat{C}_k$ and γ is a regularization parameter, usually choosen by a trial and error approach.

3.2 Parametric Density Estimation: A Numerical Example

Let us consider a 2-D data set consisting of the RMS amplitude of the seismic and the infrasonic tremor recorded during some Lava Fountain (LF) episodes that occurred at Mt Etna during 2011–2015. Recorded signals were sampled with a 10 min sampling rate and related with the observed volcanic activity, as shown in Fig. 3.

In this example the unknown state x vector is the volcanic activity observed in the summit area, which was represented as a categorical variable $x \in \Omega = [0,1,2]$, while z is a real valued vector consisting of the pairs of levels of RMS seismic and infrasonic tremor. While for this application the ultimate goal is to train a classifier which must be able to classify the status x of the volcanic activity, in this example the aim is to

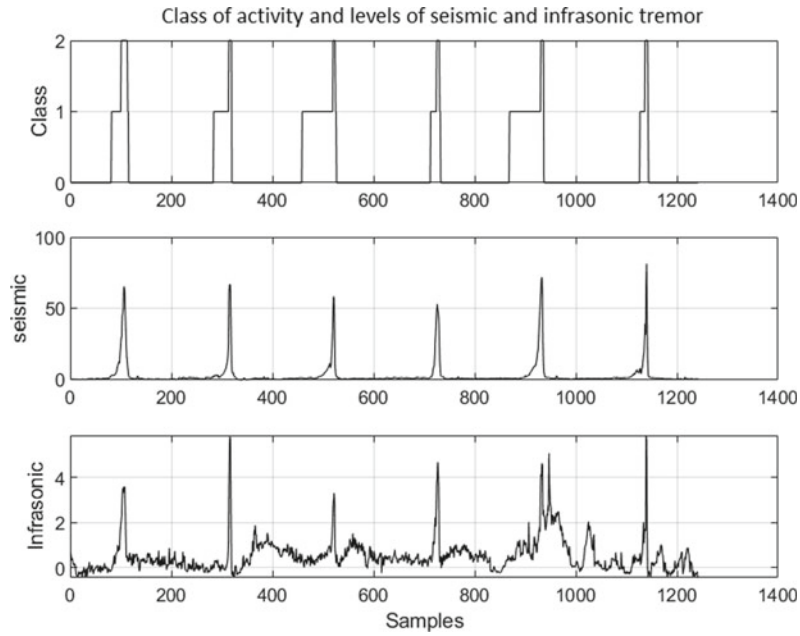


Fig. 3 Volcanic activity at Mt Etna and related RMS amplitudes of seismic and infrasonic levels

Table 1 Parameters of the gaussian function densities

K	$\hat{\mu}_k$	$\hat{C}_k$
1	[0.1844 0.6153]	$\begin{bmatrix} 0.2247 & -0.1122 \\ -0.1122 & 0.9270 \end{bmatrix}$
2	[6.6341 0.5990]	$\begin{bmatrix} 0.0172 & -0.0611 \\ -0.0611 & 1.0067 \end{bmatrix}$
3	[44.8693 3.1609]	$\begin{bmatrix} 0.0072 & -0.0289 \\ -0.0289 & 0.3543 \end{bmatrix}$

estimate the conditional probabilities $p(z|x)$ using the data set shown above. In order to improve the readability, the categorical values of the x vector are represented as Q, S and P instead of 0, 1 and 2, i.e. $x \in \Omega = [Q, S, P]$, where the symbols stand for Quite, Strombolian and Paroxysmal activity. Assuming a gaussian distribution of z values, for each class the parameters μ_k and C_k estimated by using expressions (9) and (10) are reported in Table 1, while the gaussian densities computed by using expression (8) are shown in Fig. 4. In this example we have almost manually performed the calculation in order to illustrate the use of formulas reported in this section, but the interested readers can easily find user friendly tools to automate all these calculations, as mentioned in Sect. 8.

3.3 A Non-parametric Approach

Non-parametric methods are learning methods for which prior knowledge about the functional form of the conditional probability distributions is not available or is not used explicitly. Despite the name, which could be interpreted in the sense that no parameters are involved, it is to be stressed that non-parametric approaches usually require a number of parameters larger than the parametric ones. Indeed, the term *non-parametric* must be interpreted in the sense that model parameters are not the ones of standard conditional distributions. The use of non-parametric methods allows to handle a wide range of situations when the use of standard density functions is not realistic.

One of most popular non-parametric approaches is the so-called Parzen approach (Parzen 1962), which is summarized in this section following the description given by (Lei et al. 2017).

In order to estimate $p(z|\omega_k)$ for arbitrary z, let us split the original data set T_s into K subsets T_k, as indicated in (5). Furthermore, let us partition the measurement space into a finite number of disjoint regions R_i, called bins, and count the number of samples that fall in each of these bins.

It is trivial to understand that indicating as $N_{k,i}$ the number of samples with class ω_k that fall within the i_{th} bin, then the probability density within the i_{th} bin can be estimated as:

$$P(z|\omega_k) = \frac{N_{k,i}}{Volume(R_i)N_k}, z \in R_i \qquad (12)$$

The Parzen densities obtained by using the data set considered in the previous Sect. 3.2 are shown in Fig. 5. In order to build this numerical example we have considered the PRTools (Duin et al. 2017) a suitable library running in the MATLAB framework. In particular, in this case we have used the function *parzenm,* which, given a labeled dataset, allows to compute the corresponding Parzen distribution. The graphical representation was obtained by using the *scatterd* function which allows us to represent scatter plots in 1-D, 2-D or 3-D.

3.4 Estimation of the Prior Probabilities

In order to apply the Bayes rule (4) which allow to obtain the posterior probability densities $p(\omega_k|z)$, it is necessary to know not only the prior probability densities $p(z|\omega_k)$, that we have learned in the previous sections, but also the prior probabilities $P(\omega_k), k = 1, \ldots K$. To this purpose, it

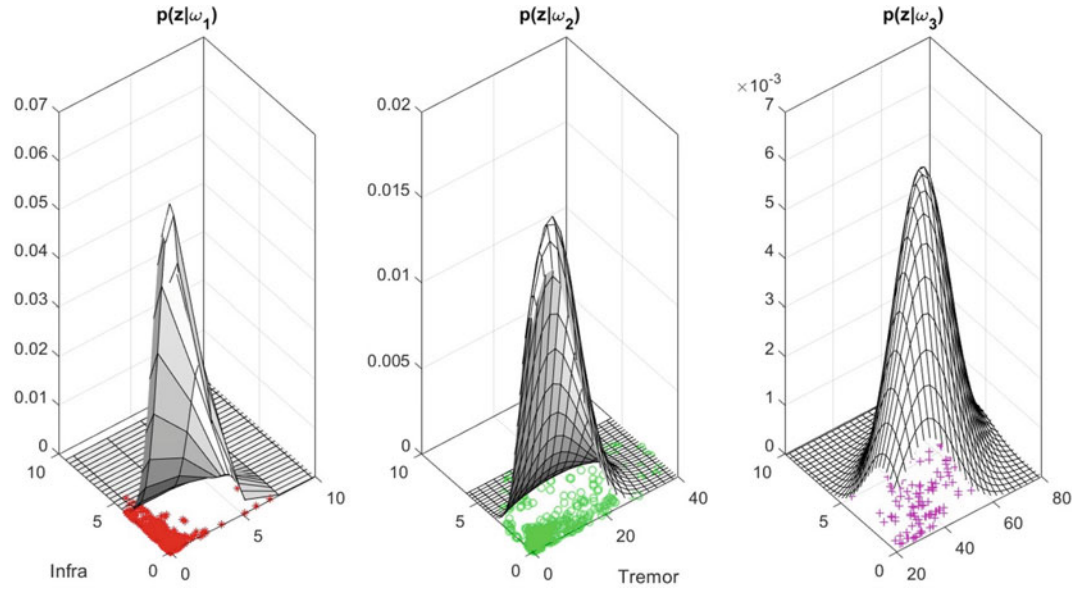


Fig. 4 Probability densities $p(z|\omega_k)$ estimated by using the parametric Gaussian approach, for the 3 classes, together with scatter plots of the seismic and infrasonic tremor samples. Here ω_1, ω_2, ω_3 stand for classes Q, S and P, respectively, while the measurement vector z has two components, referred to as seismic and Infrasonic tremor

is usually assumed that unbiased estimation of $P(\omega_k)$ can be computed with the following intuitive expression:

$$P(\omega_k) = \frac{N_k}{N_s} \tag{13}$$

4 Supervised Classification

In supervised classification, the problem is to assign a class label $\omega \in \Omega = [\omega_1, ..., \omega_K]$, which represent the state of the system associated with a measured vector $z \in R^N$. An example of classification problem of interest in active volcanic areas is that the state vector x assumes categorical values such as *Q, S, P*. Similar problems has been tackled by various authors, such as (Cassisi et al. 2016, see Sect. 3 in Chapter "Machine Learning Applications in Volcanology and Seismology"), Cannavò et al. 2017; Hajian et al. 2019, see Sect. 8 in Chapter "Machine Learning Applications in Volcanology and Seismology"), (Nunnari 2021, see Sect. 8 in Chapter "Machine Learning Applications in Volcanology and Seismology"). Another popular problem in seismology and in volcano seismology, refers to the classification of seismic waveforms, as explained in Sect. 2 in Chapter "Machine Learning Applications in Volcanology and Seismology".

4.1 The Bayes Minimum Risk Classification

It is assumed that the prior probabilities $P(\omega_k)$, $k = 1, ..., K$ are available and, since classes are supposed mutually exclusive, Eq. (3) holds. The sensory system produces vectors $z \in R^N$ of observations. As usually done in above sections, $p(z)$ indicates the unconditional density associated with z and $p(z|\omega_k)$ the conditional density of z associated with samples of the ω_k class. The problem is to estimate the so-called decision function $\hat{\omega}(\cdot)$ that maps the measurement space on to the set of possible classes Ω, that is $\hat{\omega}(\cdot)$: $R^N \rightarrow \Omega$.

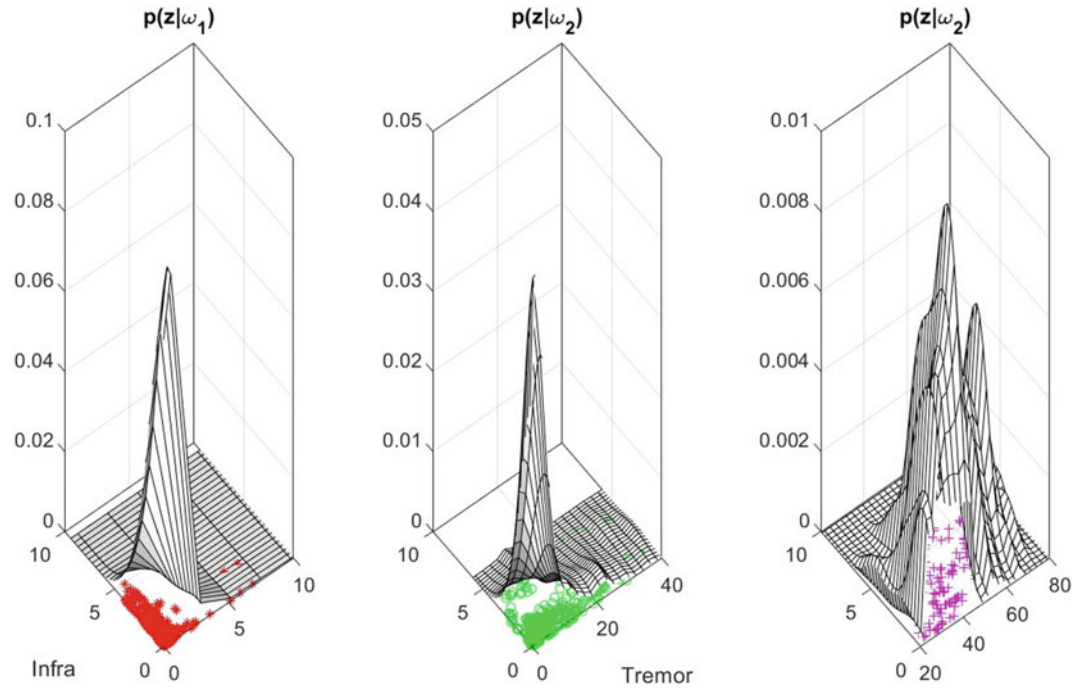


Fig. 5 Probability densities $p(z|\omega_k)$ estimated by using the non-parametric Parzen approach, for the 3 classes, together with scatter plots of the seismic and infrasonic tremor samples. Here ω_1, ω_2, ω_3 stand for classes Q, S and P, respectively, while the measurement vector z has two components, referred to as seismic and Infrasonic tremor

For a Bayes classifier, a cost function $C(\hat{\omega}(\cdot)|\omega_k)$ is usually defined which expresses the cost that is involved when the class assigned to an object is $\hat{\omega}(\cdot)$ while the true class is ω_k. The function $C(\hat{\omega}(\cdot)\ |\omega_k)$ is expressed by a square matrix of dimension K. Of course, the elements of the main diagonal of this matrix are usually set to zero, while elements of the off-diagonal are positive. For some applications the cost may be negative, since negative cost implies a profit. Together with the cost, the conditional risk of assigning a class $\hat{\omega}(\cdot)$ to a measurement sample $\boldsymbol{z}$, while the true class is ω_k, is formally expressed as

$$R(\hat{\omega}_i|z) = E[C(\hat{\omega}_i|\omega_k)|z] = \sum_{k=1}^{K} C(\hat{\omega}_i|\omega_k)P(\omega_k|z) \quad (14)$$

Integrating Eq. (14) over the whole measurement space is possible to evaluate the overall risk associated with the estimation function $\hat{\omega}(z)$. A Bayes minimal classifier is defined as the one $\hat{\omega}_B(z)$ such that:

$$\hat{\omega}_B(z) = \hat{\omega}_i : R(\omega_i|z) \leq R(\omega_j|z),\ i,j = 1,...,K \text{ i.e. :} \quad (15)$$

$$\hat{\omega}_B = \arg min_{\omega\in\Omega} R(\omega|z) \quad (16)$$

The following expression (17):

$$\hat{\omega}_B = \arg \min_{\omega\in\Omega} \sum_{k=1}^{K} C(\omega|\omega_k)p(z|\omega_k)P(\omega_k) \quad (17)$$

is referred to as the Bayes classification or minimum risk classification rule.

Another possible choice for the cost function is the following:

$$C(\hat{\omega}_i|\omega_k) = 1 - \delta(i,k) \quad (18)$$

being $\delta(i,k)$ the Kronecker delta function. With this choice the conditional risk simplifies to:

$$R(\hat{\omega}_i|z) = \sum_{k=1,\ldots K, k\neq i}^{K} P(w_k|z) = 1 - P(w_i|z) \tag{19}$$

4.2 An Example of Bayesian Minimum Risk Classifier

Let us consider the problem of classifying the volcanic activity into three classes, referred to as Q, S and P based on measures of the seismic and infrasonic tremor, already previously mentioned. For this example, the cost function was set as follows:

$$C = \begin{pmatrix} 0 & 0.07 & 0.07 \\ 0.07 & 0 & 0.07 \\ 0.07 & 0.07 & -0.02 \end{pmatrix}$$

The negative value assigned to the classification cost attributed to the P class (corresponding to paroxysmal activity) aims to emphasize that this class takes on greater importance than the others remaining ones. With this cloice the boundary lines of a minimum risk classifier implemented in MATLAB by using the PRTools, are shown in Fig. 6. In the figure, samples belonging to the Q, S and P classes are shown by using different symbols.

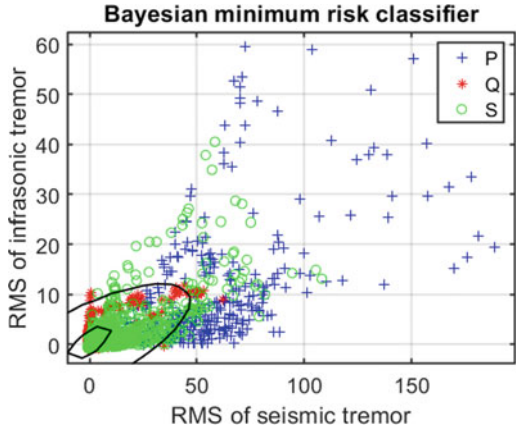


Fig. 6 Classification of volcanic activity using a Bayes classifier with minimum risk. The classification was performed assuming the seismic and the infrasonic tremor as features

It must be stressed that Fig. 6, as well as other similar ones reported trough this chapter, does not express the goodness of the classifier, but only wants to show which are the boundary lines that the classifier has estimated by using the training set. In order to objectively evaluate the performance of the classifier, it is necessary to resort to the use of appropriate performance indices, as clarified in the next Sect. 5. The Matlab code used to generate the graph shown in the figure is shown in (Table 2).

4.3 Naive Bayes Classifiers

The Naive Bayes, similarly to the minimum risk Bayesian classifier described in the previous section, make use of the Bayes theorem. However, for this algorithm it is further hypothesized that the predictors (or features) are conditionally independent (the naive hypothesis), which greatly simplifies the computation of $p(z|\omega_k)$. Indeed, based on this hypothesis, indicating as $z = [z_1, z_2, \cdots, z_N]$ the random features of an observation, it is possible to compute the $p(z|\omega_k)$ as

$$p(z|\omega_k) = \prod_{j=1}^{N} p(z_i|\omega_k) \tag{20}$$

Even if this assumption is usually violated in practice, a naive Bayes classifier usually gives good performance and is rather simple to implement.

Naive Bayes classifiers assign observations to the most probable class, i.e. in other words, apply the maximum a posteriori decision rule.

Explicitly, the algorithm consists of the following steps:

1. Estimates the densities of the predictors within each class.
2. Estimate the posterior probabilities according to the Bayes rule, that is, for all $k = 1, \ldots, K$, we have:

Table 2 Matlab code to implement the Bayesian classifier with cost function by using some of the PRTools functions in the Matlab framework

```
%Suppose that a labeled dataset ztrain was prepared for training the classifier
W = qdc(ztrain); % train a bayesian classifier with quadratic boundaries
cost = [0 0.07 0.07; 0.07 0 0.07; 0.07 0.07 -0.02]; % define the cost matrix
W2 = W*classc*costm([],cost);                % apply the cost matrix
figure; scatterd(ztrain);                    % Show scatter diagram of z
plotc(W2);                                   % Add the trained classifier
classnames = ['P';'Q';'S'];
legend(classnames)
grid on;xlabel('RMS of seismic tremor');
ylabel('RMS of infrasonic tremor');
title('Bayesian minimum risk classifier')
```

$$p(\omega_k|z) = \frac{P(\omega_k)\prod_{j=1}^{N} P(z_j|\omega_k)}{\sum_{k=1}^{K} P(\omega_k)p(z|\omega_k)} \qquad (21)$$

3. Assign $x = \omega_k$, k being the index which maximizes $p(\omega_k|z)$.

A Naïve Bayes Classifier can be easily trained by using the `naivebc` PRTools function. For example, suppose that we have prepared a labeled dataset z containing as features the RMS of seismic and infrasonic tremor recorded at Mt Etna. By using, for simplicity the default parameter allowed by the considered function, the simple Matlab code shown in Table 3 allows to obtain the results shown in Fig. 7:

The figure shows that while samples of the P class are well classified, several samples attributed by the expert to the S class are classified as Q (or vice versa).

Table 3 Matlab code to implement the Naïve Bayes classifier by using the PRTools naivebc function with default parameters

```
W = naivebc(ztrain); % compute a Naïve Bayes
classifier
figure; scatterd(ztrain); % Show scatter diagram of z
plotc(W); % Perform the scatter plot
classnames = ['P';'Q';'S'];
legend(classnames)
grid on
xlabel('RMS of seismic tremor')
ylabel('RMS of infrasonic tremor')
title('Naive Bayes classifier')
```

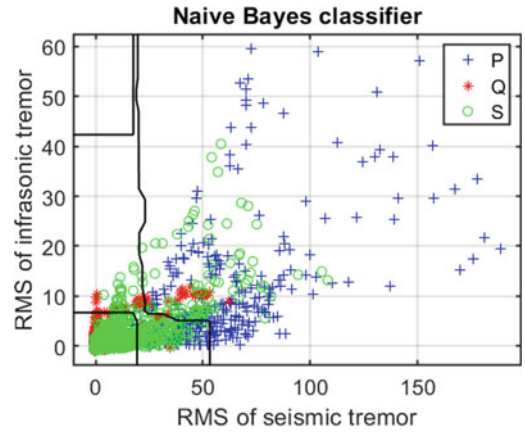


Fig. 7 Classification of volcanic activity using a Naïve Bayes classifier

4.4 Parzen Classifiers

Differently from Bayes classifiers, which usually assume standard conditional distribution densities, the Parzen estimation is based on the idea of building $p(z|\omega_k)$ without any assumption on the kind of density function. Instead, it is assumed that each sample $\boldsymbol{z_j}$ of the training set contribute to its building in a restricted neighbor of z_j, by a function $h(\cdot)$: $R^+ \rightarrow R^+$, positioned at z_j. A similar function is referred to as a kernel function. The final estimate is obtained summing the contributions obtained over the whole training set T_k. Another basic ingredient of the Parzen approach is a distance measure $\rho(z, z_j) \in R^N$, used to build a compound function $h(\rho(z, z_j))$, which must satisfy the following constraints:

- $h(\rho(z, z_j)$ must have a maximum when $z = z_j$, which of course occurs when $\rho(z, z_j) = 0$;
- it must be monotonically decreasing when $\rho(\cdot,\cdot)$ increases;
- it must be normalized to one.

With all these requirements, it will be possible to build the estimate for the conditional density $p(z|\omega_k)$, as expressed by Eq. (22).

$$p(z|\omega_k) = \frac{1}{N_k} \sum_{z_j \in T_k} h\big(\rho(z, z_j)\big) \tag{22}$$

In other terms, the kernel $h(\rho(\cdot,\cdot))$ can be regarded as a function that interpolates between the samples of the training set. Popular choices for the $\rho(\cdot,\cdot)$ and $h(\rho)$ are the following:

$$\rho(z, z_j) = \sqrt{(z - z_j)^T C^{-1} (z - z_j)} \tag{23}$$

$$h(\rho) = \frac{1}{\sigma_h^N \sqrt{(2\pi)^N |C|}} \exp\left(-\frac{\rho^2}{2\sigma_h^2}\right) \tag{24}$$

As concerning the C matrix, it must be symmetric and positive defined. In particular, the parameter σ_h in (24) is the one which controls the area around the interpolation sample z_j. It can be chosen small when the number of samples in the dataset is large. As concerning the C matrix which is also involved in expression (24), a suitable choice is to set $C = C_k$, where C_k is the covariance matrix, determined according to (10).

An example of a Parzen classifier obtained by using the same data set as in the previous sections, by using the *parzenc* function of PRTools with default parameters, is reported in Fig. 8. It can be seen, as compared to the Naive Bayes classifier shown in the previous Fig. 7, that the boundaries lines are more irregular.

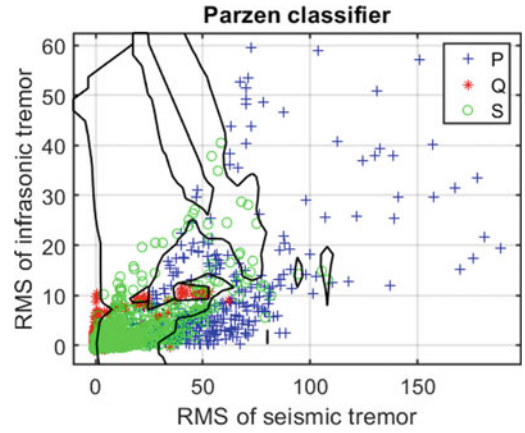


Fig. 8 Classification of volcanic activity by using a Parzen classifier

4.5 K-Nearest Neighbor (KNN) Classification

For the KNN algorithm, similarly to the Parzen algorithm, the conditional density $p(z|\omega_k)$, is based on the contribution of individual samples of the training set over a restricted neighbor of $\mathbf{z}$. But here the estimation method is based on the idea of considering for each sample z an hypersphere with volume V, centered at z, in such a way the number of samples of the training set T_k inside the volume is exactly a prefixed number κ. It is obvious that this radius depends on the position $\mathbf{z}$ in the measurement space. Indicating as $V(z)$. the volume around the generic sample z, the conditional density will be estimated according to the following expression:

$$\hat{p}(z|w_k) =_{N_k}^{\mathrm{K}} V(z) \tag{25}$$

Different choices of κ can be done for each class, A suitable choice could be to set $\kappa = \sqrt{N_k}$, where, as usual, N_k indicates the number of samples in the sub-training set T_k. After deciding how to estimate the conditional density $p(z|\omega_k)$ the development of the classifier is straightforward. Indeed, the estimation function can be written as:

$$\hat{\omega}(z) = \omega_k, k = \arg\max \hat{p}(z|\omega_i)\hat{P}(\omega_i) \tag{26}$$

which simplifies to:

$$\hat{\omega}(z) = \arg\max k_{i\,i=1,K} \tag{27}$$

The interpretation of (27) is simple: a new vector $\mathbf{z}$ will be attributed to the class which exhibits the greater number of samples in its neighborhood.

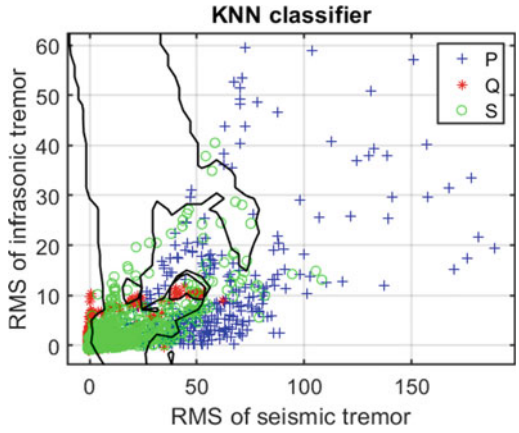


Fig. 9 Classification of volcanic activity by using a KNN classifier

The application of a KNN classifier implemented by using the *knnc* function of the PRTools, with the default parameters, is shown in Fig. 9.

It is possible to see that dealing with the KNN classifier builts quite irregular boundaries among the classes contained in the training set. But, once again we stress that this is non a guarantee that the classifier will be able to generalize the learned rule.

4.6 Classification Based on Discriminant Functions

Discriminant functions are functions $g_k(z)$, $k = 1, ...,K$, that are used in a decision function as follows:

$$\hat{\omega}(z) = \omega_n, n = \arg \max_{k=1,\cdots K} g_k(z) \tag{28}$$

Clearly, if we regard $g_k(z)$ as the posterior probabilities $p(\omega_k|z)$, the decision function becomes a Bayes decision function with a uniform cost. Since the posterior probabilities are unknown elements of a classifier, it is reasonable to replace the probabilities with some predefined discriminant functions $g_k(z)$ and estimate its parameters according to a labeled training set. In the simplest case, the samples in the training set can be satisfactorily classified by using linear decision boundaries. If this is the case, the discriminant functions assume the form

$$g_k(z) = w_k^T z + w_k \tag{29}$$

which is referred to as a linear discriminant function. Obviously, in this expression w_k is an unknown parameter vector of appropriate dimension and z the samples of the training set that will be considered, together with the associated labels, to estimate w_k. Since the problem is linear in the unknown vector parameters can be estimated by using the popular Least Square approach.

In order to simplify the notation an augmented measurement vector y is usually introduced, defined as $y = [z,1]^T$. In such a way the discriminant functions become $g_k(y) = \mathrm{w}^T{}_k\, y$, thus embedding in the vector $\mathbf{w}_k$ an extra element w_k. Furthermore, this strategy helps for the generalization of this kind of discriminant function. For instance, it is trivial to understand that assigning to the augmented measurement vector y the form:

$$y(z) = \left[z, 1, z_0^2, z_1^2, \cdots z_{N-1,}^2, \cdots, z_{N-1} z_N\right] \tag{30}$$

Where z is the generic observation vector of dimension N and z_i its components, the corresponding $g_k(y) = w_k^T y(z)$. gives a non linear classifier whose boundary regions will be a quadratic polynomial. The unknown vector of parameter $\mathbf{w}_k$, can be obtained by performing iterative algorithms. The application of a linear discriminant classifier to classify the volcanic activity, implemented by using the *fisherc* function of the PRTools is shown in Fig. 10.

The figure shows, compared to the classifiers shown above, boundary lines which are simple straight line segments and therefore easy to interpret.

4.7 The Support Vector Classifier

The support vector is a kind of classifier whose main feature is that of finding boundary lines that

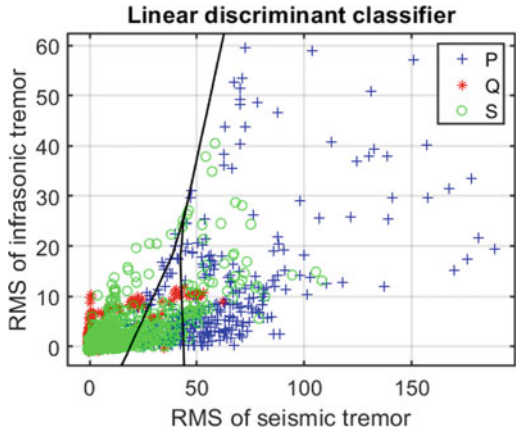


Fig. 10 Classification of volcanic activity by using a linear discriminant classifier

separates the classes with maximal margin. In order to understand the main features of this approach, let us assume that samples belong to two classes linearly separable. The margin is defined as the width of the largest gap not containing samples, that can be drawn around the decision boundary as shown in Fig. 11.

Formally, this can be expressed as follows. Assume we have training samples z_n, $n = 1,\ldots,$ N_S, and for each sample a label $c_n \in [1, -1]$, indicating the two classes to which the sample belongs. Then, we search a linear classifier $g(z) = w^T z + b$ such that $w^T z_n + b \geq 1$ if $c_n = +1$ and $w^T z_n + b \leq -1$ if $c_n = -1$. It can be demonstrated that the square of the margin is inversely proportional to the inverse of the product $\|w^2\| = w^T w$. Therefore, minimizing $\|w\|^2$,

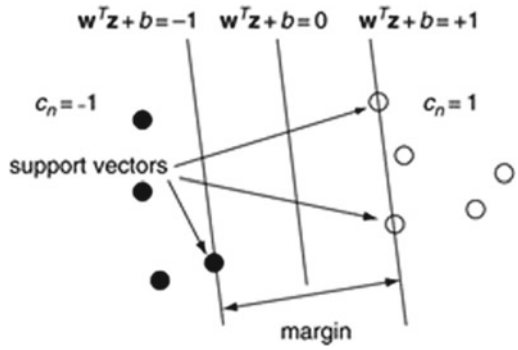


Fig. 11 Support vector classifier

it results in maximizing the margin. This task can be performed by using the Lagrange multipliers α_n. Avoiding mathematical details, in is possible to demonstrate (see Lei et al. 2017) that the problem solution can be configured as a two-step search problem which involve the following expression:

$$L = \frac{1}{2}\|w^2\| + \sum_{n=1}^{N_s} \alpha_n \left(c_n \left[w^T z_n + b\right] - 1\right), \alpha_n \geq 0 \tag{31}$$

Indeed, L should be minimized with respect to w and b, and maximized with respect to α_n. This optimization problem can be performed with the help of standard algorithms. Furthermore, it is also possible to search for non-linear boundaries, involving discriminant functions of the form:

$$g(z) = \sum_{n=1}^{N_s} K(z, z_n) \tag{32}$$

where $K(\cdot,\cdot)$ is a kernel function. In particular, kernel functions, known as radial basis functions, of the form:

$$K(z_n, z_m) = exp\left(-\frac{\| z_n - z_m \|^2}{\sigma^2}\right) \tag{33}$$

are often considered. Advantages of using SVMs for classification problems are that the model parameters can be found using standard optimization algorithms and non-linear boundaries can be used without increasing the computational effort. However, it is necessary to stress that the SVM approach usually requires high computational effort, due to the fact that the problem complexity is of the order of the number of samples. Another drawback is that the classifier is basically a two-class classifier. Therefore, in order to obtain a multi-class classifier, it is required to implement a number of two class classifiers as the number of classes to be discriminated against. For this reason when considered for multiclass problems, the SVM is usually used in conjunction with other approaches, such as the Error-Correcting Output Codes

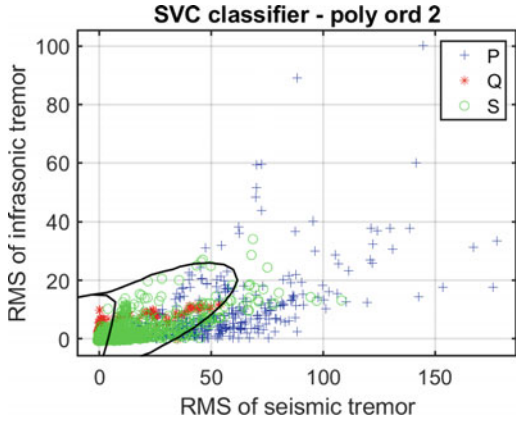


Fig. 12 Support vector classifier with quadratic kernel function to classify the volcanic activity based on RMS of seismic and infrasonic tremor

(ECOC), presented in the next Sect. 4.10. The application of a SVM classifier with a quadratic kernel function to classify the volcanic activity, implemented by using the *svm* function of the PRTools is shown in Fig. 12.

It is possible to appreciate how, thanks to the use of quadratic kernel functions, very smooth boundary lines are obtained.

4.8 Decision Trees

Decision Trees (DTs) are non-parametric supervised methods used for both classification and regression. The key idea underlying tree-based methods is that of partitioning the feature space growing a tree like structure. This consists of two entities, referred to as decision nodes and leaves, respectively. The decision nodes are nodes where each feature is splitted based on some future values, while leaves are the decisions or the final outcomes. In the classification tree the decision is a categorical value while in the regression tree it is a real value. Building a tree starts by choosing a feature to associate with the root node. This choice is made using criteria that refer to the Information Theory, such as for the information gain *IG*, to understand which it is useful to introduce the concept of Entropy associated to a dataset.

Suppose we have a data set in which K data classes are recognized. The Entropy E of the dataset S is defined according with the following expression:

$$E(S) = \sum_{i=1}^{K} -p_i \log_2 p_i \tag{34}$$

where p_i represents the probability that samples belong to the class i-th.

It is easy to recognize that in the simple case when $K = 2$, if all the samples belongs to the same class, say for instance the class 1, then we have $p_1 = 1$, $p_2 = 0$ and $E(S) = 0$, which expresses that the dataset is completely homogeneous. A similar dataset is usually referred to as *pure*. On the other hand, if samples in the dataset are mixed with equal probability, i.e. $p_1 = p_2 = 0.5$, then we have $E(S) = 1$, which indicate the maximum disorder, or in other terms the degree of *purity* is the lowest. The concept of purity of a dataset allows, in turn, to define the concept of Information Gain *IG*, which can be used to decide what advantage can derive from the division of the dataset into two subsets S_1 and S_2, using a given feature F. Indeed, still resorting to the concept of entropy of a dataset, the following expression is defined as Information Gain: $IG(S) = E(S_1) - E(S_2)$. For the purposes of the calculation of $E(S_2)$, it must be bearing in mind that after the split created by applying the feature F, the dataset will be usually partitioned into a set n of partitions and therefore $E(S_2)$ will be computed by using the following expression:

$$E(S_2) = \sum_{i=1}^{n} -w_i E(p_i) \tag{35}$$

where the coefficients w_i are the weights corresponding to each of the n sub-partitions, in proportion of the samples inside. Therefore, choosing the best feature to perform a partition, the choice is performed considering the one which allows the maximum $IG(F)$. It is trivial to understand that in the limit case when $IG(F) = 0$, there are no advantages to performing the split.

It is to be stressed that expressions considered above assume that we are working with categorical features. However, this approach can easily be extended to the case of real valued features. Indeed, in this case the feature values can be divided into class levels and therefore the corresponding *IG* can be computed. In this case, the best thresholds for defining the levels are those that guarantee the highest I*G* values. Furthermore, it should be pointed out that *IG* is not the only criterion adopted to perform the splitting in the building of a decision tree. Other popular criteria are for example the Gini index, the Deviance, and several others. In particular:

– The Gini's impurity or Gini's Diversity Index (gdi), is defined as:

$$gdi = 1 - \sum_i p_i^2(i) \tag{36}$$

where the sum is over the classes at the splitting node, and p_i is the observed fraction of classes with classes that reach the node. A node with just one class (a pure node) has a Gini index 0; otherwise the Gini index is positive.

– The Deviance, defined as:

$$D = \sum_i p(i) \log_2 p(i) \tag{37}$$

where p_i is defined as for the Gini index.

A very simple example of DT grown to classify volcanic activity into three classes Q, S and P, based on the level of the seismic tremor is shown in Fig. 13.

This figure was obtained by using the Python scikit-learn package (https://scikit-learn.org/stable/), mentioned in Sect. 8.2. In order to simplify the figure the maximum number of allowed splits was limited, but normally DT trees are not so easy to visualize and interpret.

Some advantages of decision trees are:

– Since DTs can be visualized, they can be simple to be interpreted, provided that the number of nodes is limited.
– DTs usually require a few efforts for data preparation. For instance, a data normalization phase is not required.
– The cost of using DTs is logarithmic versus the number of data points used to train the tree.

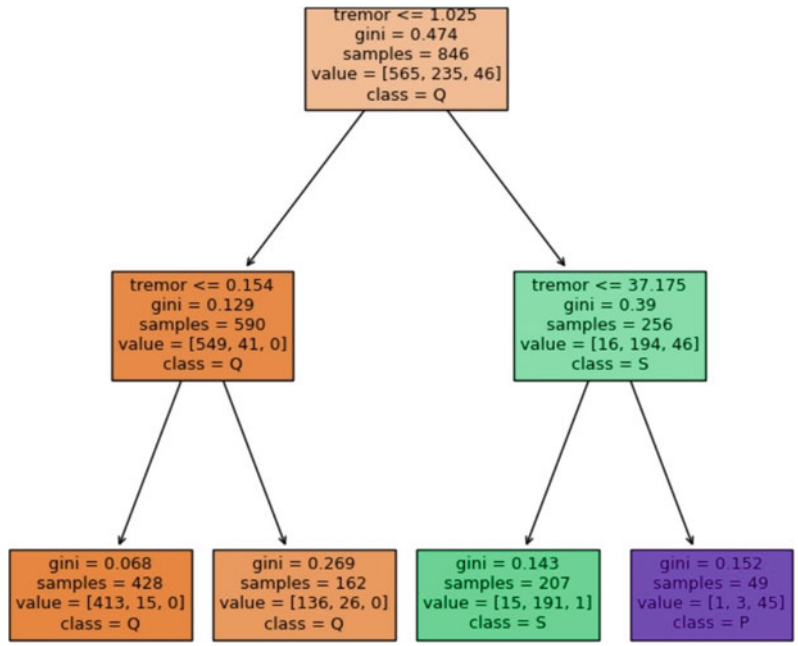


Fig. 13 A sample of decision tree classifier used to classify volcano activity through tremor data

– DTs are able to handle both numerical and categorical data and multi-class problems.

The disadvantages of decision trees include:

– Decision-tree classifiers are subject to a higher level than others approaches, to the problem of overfitting. Therefore, the user can grow a DT which is able to perfectly classify all the samples of the training set but it does not correctly work with the test set.
– Decision trees can be *unstable* in the sense that small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble approach, as explained in the next section, devoted to combining models by using the Boosting and Bagging approaches.

The application of a DT classifier implemented by using the function *treec* of the PRTools with default parameters is shown in Fig. 14.

It can be seen that the boundaries lines are irregular and often, in the attempt to classify isolated observations, the boundaries are difficult to interpret on the basis of physical principles. This can be interpreted as an indication of overfitting.

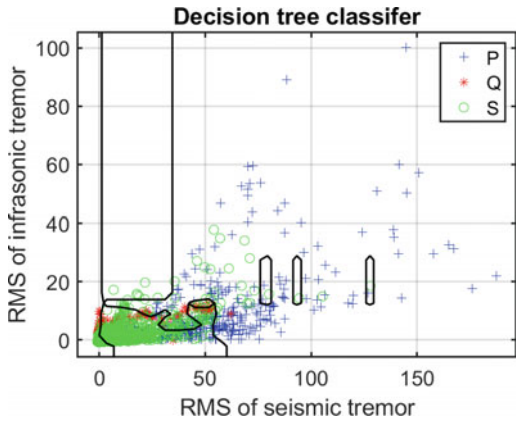


Fig. 14 A sample of decision tree classifier used to classify volcano activity through infrasonic and seismic tremor data

4.9 Combining Models: Boosting and Bagging

In earlier sections, we have explored a range of different models for solving classification problems. However, it is often found that improved performance can be obtained by combining together multiple models. In other terms, a more robust alternative in solving a given classification problem is to use different classifiers and make predictions using some criterion averaging the predictions made by each individual classifier. Such a combination of models is usually referred to as a *committee*.

4.9.1 Boosting

The supervised classification approaches reviewed in the previous sections are all aiming to train a strong classifier directly. Instead, the boosting algorithm offers an indirect way of converting a series of weak classifiers into a strong one. The term *weak* is referred to a classifier that performs even only slightly better than a random one. In other terms, boosting refers to a general method of obtaining an accurate classifier by combining rough and moderately inaccurate classifiers. There are several boosting algorithms, such as the popular *Adaboost*, which is the acronym of Adaptive Boosting. Originally designed for solving classification problems, boosting was further extended also for regression. In order to justify the improvement make possible by a boosting approach, let us consider for example a set of k regression models, each making an error $E(\epsilon_i)$, referred to the i-th example. Suppose that the errors follows a normal distribution with zero-mean, variances $v = E\left(\epsilon_i^2\right)$ and covariance $c = \epsilon_i \epsilon_j$. Then it is possible to demonstrate that the expected error of the ensemble predictor is:

$$E\left[\left(\frac{1}{k}\sum_i \epsilon_i\right)\right] = \frac{1}{k}v + \frac{k-1}{k}c \qquad (38)$$

This means that in the case where the errors are perfectly correlated and $c = v$, the mean squared error reduces to v and averaging the model does

not give any advantage. Instead, in the case where the errors are perfectly uncorrelated and $c = 0$, the expected squared error of the ensemble decreases as the reciprocal of the number of classifiers, thus improving the performances with respect to the individual classifiers.

The application of a *Adaboost* classifier implemented by using the *adaboostc* function of the Matlab PRTools, with default parameters and choosing as weak classifier a decision tree, implemented by the *treec* PRTools function, is shown in Fig. 15.

Instead, if a quadratic Bayesian classifier was used as a weak classifier, the *adaboost* algorithm gives the results shown in Fig. 16.

In this last case the boundary lines, even if not perfectly quadratic, since obtained averaging an enseble of quadratic Bayesian classifiers, looks quite different then the ones in Fig. 15 and perhaps more realistic.

4.9.2 Bagging

Bagging is a term obtained combining the words *bootstrap* and *aggregating*, which indicate a technique for reducing the generalization error by combining several models. The idea is to train several different models separately, then each individual model will contribute to the classification process by expressing a vote. In order to understand the underlying principle, let us suppose the original data set consists of N_S data

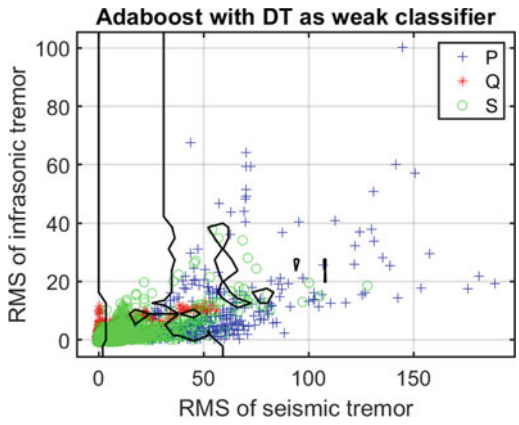


Fig. 15 A sample of a Adaboost classifier choosing a decision tree as a weak classifier, used to classify the volcano activity through infrasonic and seismic tremor data

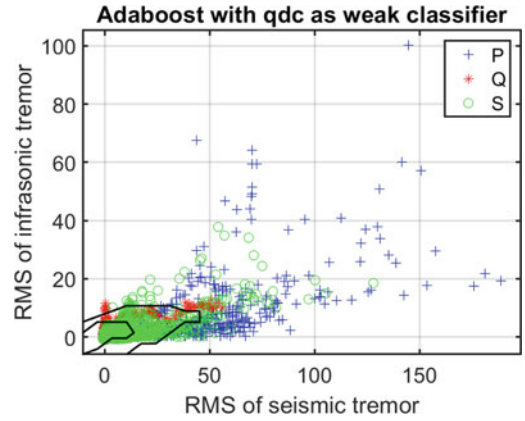


Fig. 16 A sample of a Adaboost classifier choosing a quadratic Bayesian classifier as a weak classifier, used to classify the volcano activity through infrasonic and seismic tremor data

points $Z = z_1, \cdots z_{N_s}$. Starting from Z, others dataset, generically indicated as Z_B, are generated, by using the so-called bootstrap technique. This consists in drawing N_S points at random (one at time) from Z, with replacement, so that some points in Z may be replicated in Z_B, whereas other points in Z may be absent from Z_B. Now, suppose that we want to solve a regression problem and to this purpose we generate M bootstrap dataset. By using each dataset, we train a model which predicts a value $x_m(z)$. At the end of the process, the final value corresponding to a new observation z will by computed by the following expression:

$$x_{COM}(z) = \frac{1}{M}\sum_{m=1}^{M} x_m(z) \tag{39}$$

Thanks to this averaging strategy, this kind of models are able to overcome one of the main weak point of decision trees which, as stressed in the previous Sect. 4.8 tend to overfit their training set. The popular Random Forests (Breiman 2001) or random decision forests algorithm is a kind of ensemble learning method for both classification and regression that applies the Bagging approach explained above. The application of a Random Forest classifier, implemented by using the *randomforestc* function of the PRTools, which grows by default 200

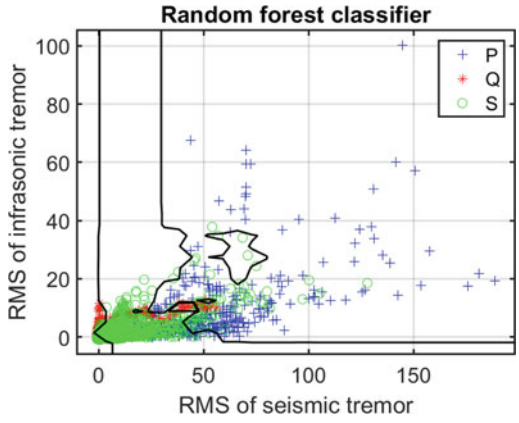


Fig. 17 A random forest classifier obtained averaging over 200 trees, used to classify the volcano activity

decision trees, for classify the volcanic activity is shown in Fig. 17.

It can be seen that, for the considered dataset it produces boundaries lines quite similar to that obtained by using the Adaboost with a DT as weak classifier, shown in Fig. 15.

4.10 Error-Correcting Output Codes (ECOC)

The Error-Correcting Output Codes (ECOC) (Dietterich and Bakiri 1995) is an ensemble method designed for multi-class classification problems (i.e. $K > 2$), by combining L binary classifiers. Therefore, this approach allows us to overcome the limitation of binary classifiers, such as for instance the SVM ones. The number L of binary classifiers to solve a given multiclass problem depends on the kind of coding considered in order to perform the final classification. Among several available coding techniques the most popular are probably the One-Versus-All (OVA), and the One-Versus-One (OVO). It is trivial to understand that the coding approach affects the number of required binary classifiers. For instance, in the case of adopting the OVO coding, the number of needed binary classifiers is $L = K(K - 1)/2$, whereas K is the number of classes.

The basic ingredients of the ECOC method are a coding design matrix M and a binary loss is function $g(\cdot,\cdot)$.

The coding design matrix is a matrix whose elements decide which classes are trained by each binary classifier, that is, how the multiclass problem is reduced to a series of binary classification problems. Each row of the coding design matrix corresponds to a distinct class, and each column corresponds to a binary classifier. In a ternary coding design, for a particular column:

- A row containing 1 directs the binary classifier to group all observations in the corresponding class into a positive class.
- A row containing − 1 directs the binary classifier to group all observations in the corresponding class into a negative class.
- A row containing 0 directs the binary classifier to ignore all observations in the corresponding class.

For instance, suppose that the goal is to classify the volcanic activity data into 3 classes, namely Q, S and P, choosing the one-versus-one (OVO) coding. In this case, the coding matrix reported in Table 4 expresses that the classifier C1 is trained by using samples belonging in Class Q and S and refer to Class Q as the positive class and Class S as the negative. In a similar way can be interpreted the tasks assigned to the classifiers C2 and C3.

A binary loss function $g(\cdot,\cdot)$, whose arguments are the class and the classification score, is considered in order to establish which is the best class to assign a new observation, once the L classifiers involved in an ECOC classifier have been trained. In more detail, the ECOC algorithm

Table 4 Coding design matrix for the reported example

Class	C1	C2	C3
Q	1	1	0
S	−1	0	1
P	0	−1	−1

assigns a new observation to the class $\hat{k}$ that minimizes the aggregation of the losses for the L binary classifiers. Different expressions can be used to compute the predicted class for the observation. For instance, in the *loss-based decoding*, the class producing the minimum sum of the binary losses over binary classifiers (Escalera et al. 2010) is computed as:

$$\hat{k} = \arg \min_{k=1,\ldots K} \sum_{l=1}^{L} \left| m_{k,l} \right| g\left(m_{k,l}, s_l\right) \qquad (40)$$

where:

- $m_{k,l}$, is the generic entry of the design matrix,
- s_l the predicted classification score for the positive class of the classifier l
- $g(\cdot,\cdot)$ the binary loss is function
- L the number of trained classifiers.

The application of the ECOC approach, implemented by using the Matlab function *fitcecoc* with the default parameters, to classify the volcanic activity at Mt Etna based on the seismic and infrasonic tremor, is shown in Fig. 18.

This ECOC classifier considers SVM classifiers with quadratic learners as binary classifiers. It can be appreciated how the algorithm produces boundaries lines among the three classes which are very clear and easily interpretable bearing in mind the classification physic of the phenomenon: low levels of both seismic and infrasonic tremor generate Q (Quite) activity, average values generate the S class (i.e. the Strombolian activity) and, finally, high seismic and infrasonic tremor generate the class P (i.e. the Paroxysmal activity).

4.11 Hidden Markov Models

A Hidden Markov Model (HMM) is a kind of state space model which describes a sequence $x(t)$ of discrete states starting at time $t = 0$, based on a sequence of measurements $z(t)$ that can only take values from a finite set. The model consists of the following ingredients:

- A finite set S containing the m states that $x(t)$ can assume.
- A set O containing the n symbols O_i that $z(i)$ can take.

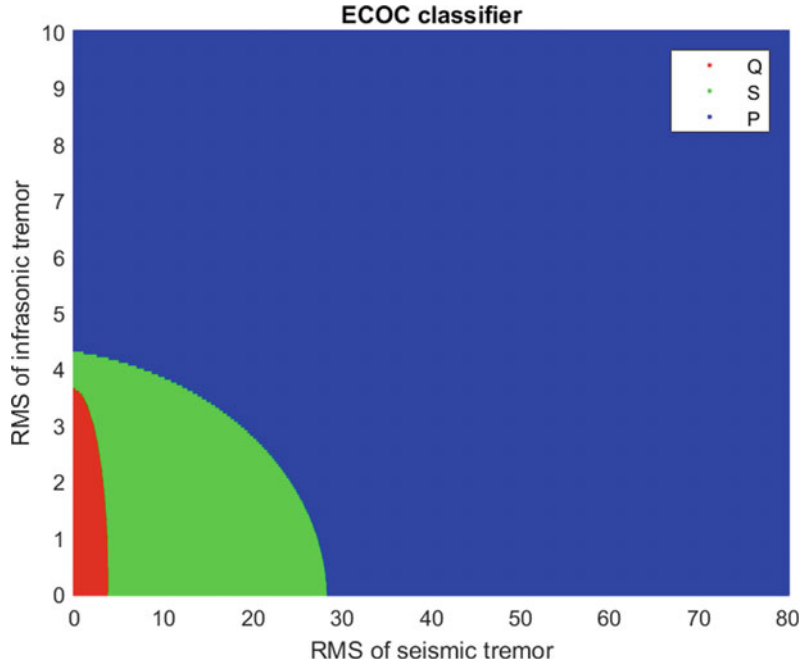


Fig. 18 An ECOC classifier, used to classify the volcanic activity at Mt Etna

- A m by n matrix A, referred to as the state transition matrix, whose generic (i, j) entry is the probability of transition from state x_i to state x_j.
- A set of possible observations, also referred to as tokens, $O = \{O_1, O_2, \ldots O_n\}$.
- An m by n matrix E, referred to as the emission matrix, whose generic entries (i, j) express the probability of emitting observation O_j given that the model is in the state x_i.
- A vector π of probabilities, where the generic entry π_i denotes the probability of starting at a given state i in the first time point. Of course the following equation must hold: $\pi_1 + \pi_2 + \ldots \pi_m = 1$.

The operation of an HMM model is schematically represented in Fig. 19.

The arrows in the diagram, usually referred to as a *trellis*, denote conditional dependencies. It is clear that the conditional probability distribution of the hidden variable $x(t)$ at time t, depends only on the value of the hidden variable $\mathbf{x}(t-1)$, which is usually referred to as the Markov property. Similarly, the value of the observed variable $z(t)$ only depends on the value of the hidden variable $x(t)$. In the standard hidden Markov models both the state space of the hidden variables and the observation variable are categorical. However, more generally, in some cases the observation variables can be continuous, typically following a Gaussian distribution. The main problems associated with a HMM model λ can be expressed as follows:

- *Evaluation*: find the probability $P(O|\lambda)$ to have a sequence of observation $O = \{O_1, O_2, \ldots O_L\}$, given a model λ. Two popular algorithms are available to compute $P(O|\lambda)$, usually referred as the forward and the backward algorithms.
- *Decoding*: given a model λ and a sequence of observations O, find the sequence S of states that maximize $P(O|\lambda)$. This task is usually performed by using the popular Viterbi algorithm.
- *Learning*: given a model λ, with unknown transition A and emission E matrices, find the set of model parameters that maximize $P(O|\lambda)$. This task is usually performed by using algorithms such as the Baum-Welch algorithm or the Baldi-Chauvin algorithm.

The application of HMM models to classify the volcanic activity at Mt Etna is given in Sect. 3 in Chapter "Machine Learning Applications in Volcanology and Seismology", following the description given by (Cassisi et al. 2016).

5 Classification Metrics for Model Validation

As mentioned in Sect. 1 in order to build a model by using supervised approaches the available data set must be divided into two or better into tre sets. In this last case these are referred to as *training, validation* and *testing* sets. A common practice consists of assigning about 70% of available data to the training set and the remaining will be assigned to the validation and to the test set, in equal parts. The validation set usually is used to halt the training when the generalization stops improving while the testing set is used exclusively to assess the model performance. Alternatively, especially when the

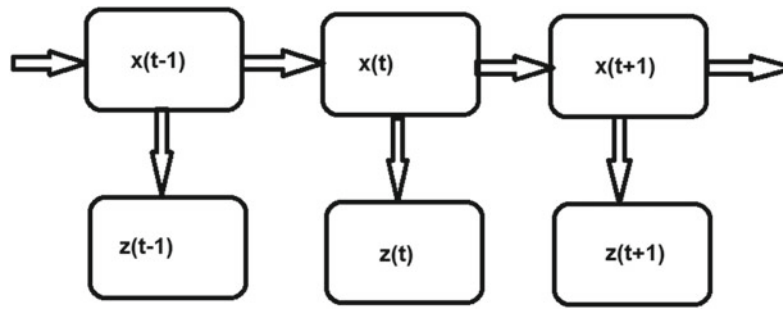


Fig. 19 The operation of a HMM model

number of data examples available is limited, a popular alternative to divide the dataset into training, validation and testing is the *k-fold cross* validation strategy. This approach consists of splitting the dataset into k groups, after shuffling the samples randomly. Then for each unique group the following steps are performed:

- A group is held out and used for testing.
- The remaining groups are used for training a model.
- The fitted model is evaluated on the testing set.
- The obtained score is then retained while the model is discarded.

This means that each sample is given the opportunity to be used in the hold out set 1 time and to train the model $k - 1$ times. At the end of the rotation process, the skill of the model is evaluated using the sample of model evaluation scores.

After a supervised classifier has been built, it is necessary to resort to objective methods to evaluate its performance. These methods in practice consist in computing some quality indices. For the readers who are interested in developing ML applications or even simply interested in reading the vast ML literature, there is the arduous task of dealing with the plethora of existing *performance indices.* A common denominator of these indices is that they refer directly or indirectly to two basic elements: the set of actual (or true) classes and the set of predicted classes. The aim of this section is to present the most popular performance indices that are used to compare these two classes of objects.

Given a classification experiment, the following rates are usually defined:

$$TPR(i) = \frac{TP(i)}{P(i)} = \frac{TP(i)}{TP(i) + FN(i)} = 1 - FNR(i) \tag{41}$$

$$TNR(i) = \frac{TN(i)}{N(i)} = \frac{TN(i)}{TN(i) + FP(i)} = 1 - FPR(i) \tag{42}$$

$$FNR(i) = \frac{FP(i)}{P(i)} = \frac{FN(i)}{FN(i) + TP(i)} = 1 - TPR(i) \tag{43}$$

$$FPR(i) = \frac{FP(i)}{N(i)} = \frac{FP(i)}{FP(i) + TN(i)} = 1 - TNR(i) \tag{44}$$

The meaning of the above indices can be expressed as follows:

- The $TPR(i)$ expresses the proportion of actual positives that are correctly classified by the model as belonging to the i_{th} class. The TPR, also referred to as *Recall* (r) ranges between 0 and 1, 1 being the best value.
- The $TNR(i)$ expresses the proportion of actual negatives that are correctly classified as not belonging to the i_{th} class. As for the *TPR*, the best values of TNR approaches 1.
- The $FNR(i)$ expresses the proportion of false negatives in the i_{th} class with respect to all actual positives in the same class. Of course, in the best case *FNR* approaches 0.
- The $FPR(i)$ expresses the proportion of false positives in the i_{th} class with respect to the total number of actual negatives in the same class. Similar to the *FNR*, in the best case *FNR* approaches 0.

These four indices are usually arranged in order to build the so-called confusion matrix, as shown, for instance, in Fig. 20.

Another useful index, is the Positive Predicted Value *(PPV)* or simply *Precision*, which, for the generic class i is defined as:

$$PPV(i) = \frac{TP(i)}{TP(i) + FP(i)} = 1 - FDR(i) \tag{45}$$

where FDR stands for False Discovery Rate. Obviously the best values of PPV and FDR are 1 and 0, respectively.

On the CM, the rows correspond to the predicted class, in the figure indicated as Output Class, and the columns correspond to the true class, also referred to as the Target Class. The

Fig. 20 Example of a confusion matrix for a three class classifier

diagonal cells indicate the number of observations that are correctly classified, while the off-diagonal cells correspond to observations incorrectly classified. The column on the far right of the CM shows, for each class, the corresponding PPV and the FDR. Similarly, the row at the bottom of the CM the TPR and the FNR, respectively.

Another popular index is the f_1-score, which represents the harmonic means between $p(i)$ and $r(i)$ and therefore defined as:

$$f_1(i) = \frac{2TP(i)}{2TP(i) + FP(i) + FN(i)} \tag{46}$$

Indicating as $r(i)$ the TPR for the i_{th} class and as $p(i)$ the PPV, the f_1 can be also written as:

$$f_1(i) = 2\frac{r(i)p(i)}{r(i) + p(i)} \tag{47}$$

Another index widely considered in literature for assessing the goodness of a classifier is the Accuracy, formally defined as

$$Accuracy = 100\frac{\sum_{n=1}^{N} I(C(x_n = y_n))}{N} \tag{48}$$

where

- $I(g)$ is a function that returns 1 if g is true and 0 otherwise
- $C(x_n)$ is the class label assigned by the classifier to the sample x_n
- y_n is the true class label of the sample x_n
- N is the number of samples in the testing set.

A useful index for evaluating the accuracy of a classifier, compensation for random hits, which can be computed starting from entries of the confusion matrix (CM), is the Cohen's Kappa, which is defined as

$$\kappa = \frac{N\sum_{i=1}^{m} CM_{ii} - \sum_{i=1}^{m} C_{itrue} - C_{ipred}}{N^2 - \sum_{i=1}^{m} C_{itrue} - C_{ipred}} \quad (49)$$

where:

- m is the number of classes,
- N is the number of samples in the testing set
- $CM_{ii}, i = 1, \cdots, m$ are the entries of the CM main diagonal
- C_{itrue} the true number of samples in class i
- C_{ipred} the predicted number of samples in class *i*.

The range of κ values is $-1 \leq \kappa \leq 1$, with *1* indicating strong agreement, -1 indicating strong disagreement, and 0 indicating chance-level agreement.

The use of these performance indices will be useful to understand the applications described in the next Chapter "Machine Learning Applications in Volcanology and Seismology" of this book.

6 Unsupervised Classification

While in supervised classification, the problem is to assign a class label $\omega \in \Omega = [\omega_1, ..., \omega_K]$, which represent the state of the system associated with a vector of measures $z \in R^N$ and therefore a dataset is represented by pairs (z, ω), in unsupervised classification, also referred as clustering, the main problem is to look for hidden structures in the set of measure vectors $z_i, i = 1, \ldots N_s$ (i.e. the set of observations). For instance, we want to know if the vectors z_i can be grouped into homogeneous classes, whose number is usually not a-priori known, but always inferable thanks to several existing algorithms, such as for instance the Silhouette index (Rousseeuw 1978). Given a data set of vectors z_i a cluster can be defined introducing the concept of similarity between the objects. Indeed, objects within a cluster are those whose degree of similarity is greater than the similarity with other objects belonging to other subsets. As concerning the degree of similarity, it is usually measured referring to the well-known concept of distance between objects, such as the traditional Euclidean distance or other definitions, such as the Mahalanobis distance (50) or the cosine distance (51)

$$d_p(z_i, z_j) = \sum_{n=1}^{N} (z_{i,n}, z_{j,n})^{\frac{1}{p}} \quad (50)$$

$$d_p(z_i, z_j) = 1 - \frac{z_i^T}{||z_i||^2 ||z_j||^2} \quad (51)$$

However, it must be stressed that distances above can be computed only in case of real valued observations. Different approaches must be considered in case the observations are ordinal or categorical.

Due to the fact that clustering is unsupervised, it is very difficult to evaluate its goodness. Indeed, different clustering methods will produce different clusters and results cannot be in general compared, unless the model is trained by using a labeled dataset, but where labels are not taken into account for the clustering, but for performance evaluation only.

Among several existing approaches, in this book we briefly present the Hierarchical clustering, the K-Means, the Fuzzy C-Means (FCM) and the Gaussian Mixture Models (GMM).

6.1 Hierarchical Clustering

Hierarchical clustering is an iterative algorithm that consists of gathering objects into larger clusters by combining objects and clusters, until all objects belong to one cluster. Therefore, objects are not necessarily assigned to a number K of classes defined a-priori. This provides more information about the structure in the dataset and shows which clusters are similar or dissimilar. In such a way the user may have an idea on how observations best aggregate and to detect

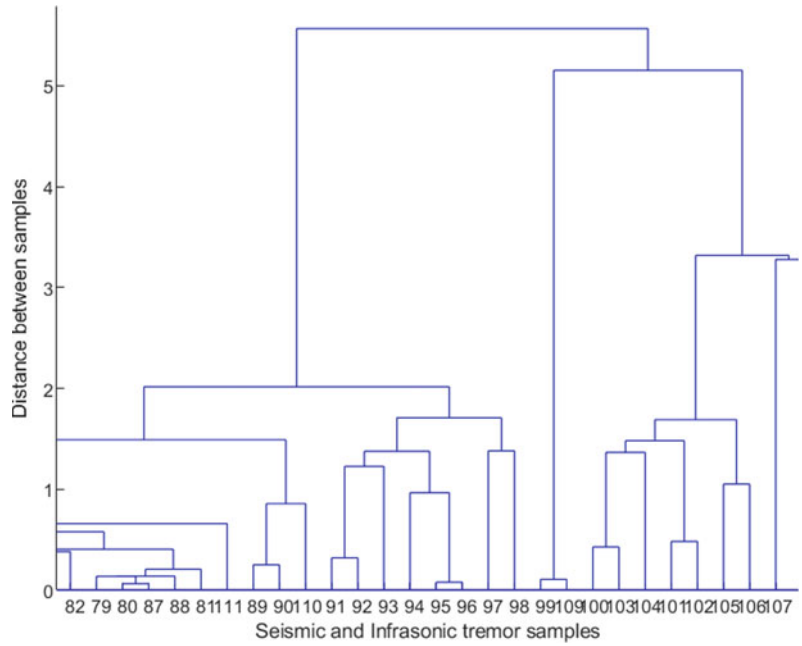


Fig. 21 Dendogram which represents the hierarchical clustering of pairs of seismic and infrasonic tremor. For clarity reasons only a portion of the dendogram is shown

anomalous clusters. The algorithm works as follows:

1. Initially each observation of the dataset is assigned to its own cluster. Therefore, the aggregation process starts having an initial number of clusters equal to the number of observations contained in the dataset.
2. The pair of observations with the shortest distance are determined and grouped into a single cluster
3. Step 2 is repeated until only one cluster is obtained.

This algorithm allows to build a dendogram such as the one shown in Fig. 21. In this case the clustered dataset consists of seismic and infrasonic tremor pairs recorded at Mt Etna. In order to make the figure understandable, only a portion of the *x*-axis is shown.

The integers in the x-axis represent the ordinal number of observations as entries of the feature matrix matrix X, while in the y-axis the distance between observations is reported. The reader may realize that it is not easy to read a dendogram with hundreds or thousand elements, but he can at least have a rough idea of the most populated groupings. The dendogram shown (as a portion) in Fig. 21 was obtained by the simple Matlab code shown in Table 5. The functions *distm* and *hclust* considered for this code belong to the PRTools package.

Table 5 The Matlab code to perform the hierarchical clustering of a feature matrix X of seismic and infrasonic tremor

```
% X is the matrix of seismic and infrasonic tremor samples
d = sqrt(distm(X));                    % computed the distance matrix
den = hclust(d,'s');                   % perform the hierarchical clustering
figure; plotdg(den);                   % plot the dendogram
xlabel('Seismic and Infrasonic tremor samples');
ylabel('Distance between samples')
```

6.2 K-Means Clustering

The K-means is a clustering algorithm based on the idea of assigning observations to classes based on their distance from the so-called cluster centers. Therefore, in this algorithm, unlike hierarchical clustering, it is necessary to identify in advance the number of classes K to be

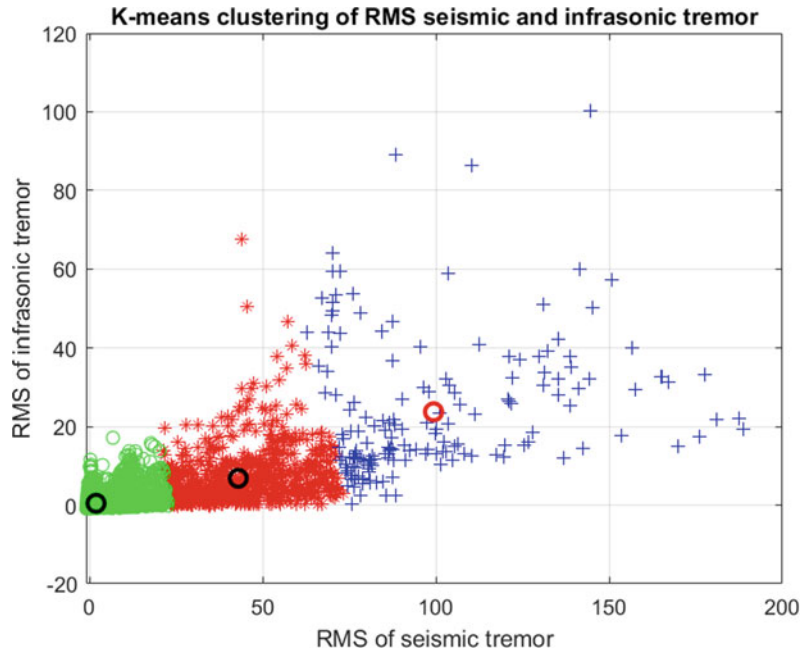


Fig. 22 Clustering the RMS of seismic and infrasonic tremor at Mt. Etna by using the K-means algorithm. Different colors indicate the three searched clusters. The circles symbols indicate the cluster centers

obtained. The algorithm can be summarized as follows.

1. Observations are initially randomly assigned to one of the clusters $k = 1,...K$.
2. The means $\mu_k, k = 1, \ldots, K$ of each cluster is computed as:

$$\mu_k = \frac{1}{N_k} \sum_{z_i \in C_k} z_i \tag{52}$$

3. Each observation z_i is then reassigned to the cluster with the closest mean μ_k.

Step 2 and 3 are repeated until the means of the clusters do not significantly change.

One of the advantages of this algorithm is its simplicity which results in a very easy implementation and very fast computation. However, due to the random choices involved, running the algorithm several times on the same dataset in some cases different results can be obtained. In more technical terms, the algorithm might converge to different local minima. Furthermore, when a high number K of clusters is set, it may happen that some clusters do not gain sufficient support and are then ignored giving a number of clusters less than K.

The application of the *K-means* algorithm to clusterize the seismic and infrasonic tremor at Mt Etna, by using the Matlab *kmeans* function is shown in Fig. 22. The very simple Matlab code executed for this purpose is shown in Table 6.

The clustering shown in Fig. 22 is easily interpretable and adheres to the physical principle that links the observed volcanic activity and the levels of seismic and infrasonic tremor.

6.3 Fuzzy c-means

The Fuzzy c-means (FCM) can be considered an extension of the *K*-means, but while in this latter algorithm each point belongs exclusively to one class, the FCM allows each data point to belong to multiple clusters, with varying degrees of membership. In more detail the FCM algorithm is based on the minimization of the following cost function:

Table 6 Matlab code for using the Matlab *kmeans* function

```
%X is the matrix containing the samples seismic and
infrasonic tremor
[cidx, ctrs] = kmeans(X, 3, 'Distance','sqeuclidean',
'Replicates',5);
plot(X(cidx =  = 1,1),X(cidx =  = 1,2),'r*', X
(cidx =  = 2,1),X(cidx =  = 2,2),'go',…
X(cidx =  = 3,1),X(cidx =  = 3,2),'b + ');
hold on
plot(ctrs(1,1),ctrs(1,2),'ko','MarkerSize',8,'linewidth',2)
plot(ctrs(2,1),ctrs(2,2),'ko','MarkerSize',8,'linewidth',2)
plot(ctrs(3,1),ctrs(3,2),'ro','MarkerSize',8,'linewidth',2)
xlabel('RMS of seismic tremor')
ylabel('RMS of infrasonic tremor')
title('K-means clustering of RMS seismic and infrasonic
tremor')
grid on
hold off
```

$$J = \sum_{i=1}^{N_s} \sum_{j=1}^{K} \mu_{i,j}^{m} ||z_i - \mu_j||^2 \tag{53}$$

where:

- N_s is the number of data points.
- K is the number of clusters.
- m is the fuzzy partition matrix exponent for controlling the degree of fuzzy overlap, with $m > 1$.
- z_i is the *i-th* data point.
- μ_j is the center of the *j-th* cluster.
- μ_{ij} is the degree of membership of z_i in the *j-th* cluster. Obviously, for a given data point, z_i, the sum of the membership values for all clusters is one.

The FCM algorithm performs the following steps:

1. Randomly initialize the cluster membership values μ_{ij}.
2. Calculate the cluster centers:

$$c_j = \frac{\sum_{i=1}^{D} \mu_{ij}^{m} z_i}{\sum_{i=1}^{D} \mu_{ij}^{m}} \tag{54}$$

and update μ_{ij} according with the following expression:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{N} \left(\frac{\|z_i - c_j\|}{\|z_i - c_k\|} \right)^{\frac{2}{m-1}}} \tag{55}$$

3. Calculate the objective function, J_m.
4. Repeat steps 2 to 4 until J_m improves by less than a specified minimum threshold or until after a specified maximum number of iterations.

The user can choose a fuzzy partition matrix exponent, indicated as m, being $m > 1$, for controlling the degree of fuzzy overlap. Usually the default choice for m is $m = 2$.

The application of the FCM algorithm to clusterize the seismic and infrasonic tremor samples considered through this chapter is shown in Fig. 23. For this purpose the *fcm* function belonging to the Matlab Fuzzy Logic Toolbox was considered, inside the code reported in Table 7.

It can be seen that clustering results shown in Fig. 23 are, at least from the visual point of view, very similar to that obtained by using the K-means algorithm. This confirms that the two algorithms are very similar, being the greatest difference in the presence of the parameter m which offers a little more chance to the user.

6.4 Mixture of Gaussians

Gaussian mixture models (GMM) are probabilistic algorithms which allow the use of independent gaussian distributions for each of the K clusters of a multidimensional dataset. In such a way the mixture probability density function assumes the form:

$$p(z) = \sum_{k=1}^{K} \pi_k N(z|\mu_k, C_k) \tag{56}$$

where $N(z|\mu_k, C_k)$ stands for the multivariate Gaussian distribution, π_k are the so-called mixing coefficients for which $\sum_{k=1}^{K} \pi_k = 1$ and $\pi_k \geq 0$. $N(z|\mu_k, C_k), k = 1, \ldots K$ are normal distribution densities with mean μ_k and covariance C_k.

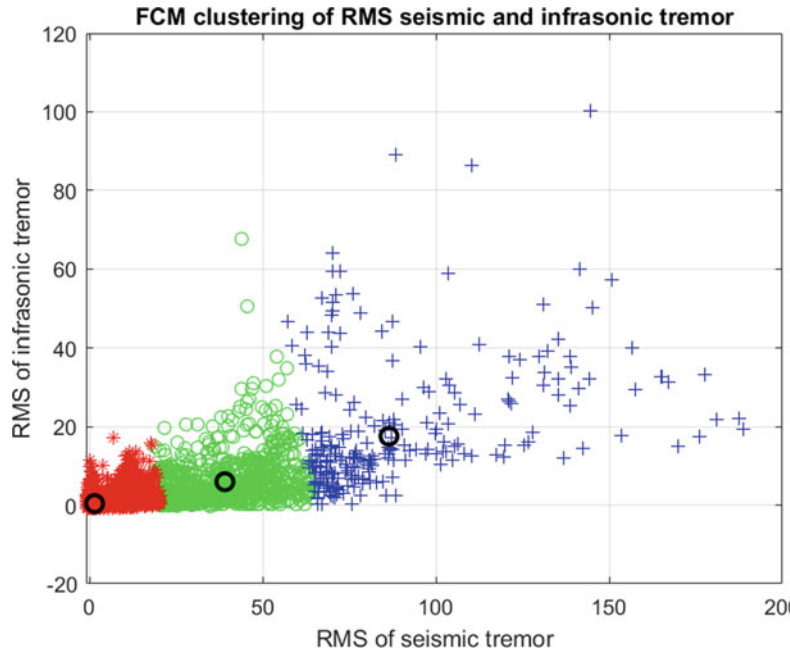


Fig. 23 Clustering the RMS of seismic and infrasonic tremor at Mt. Etna by using the FCM algorithm. Different colors indicate the three searched clusters. The circles symbols indicate the cluster centers

Table 7 Matlab code for using the Matlab *fcm* function

```
%X is the matrix containing the samples seismic and
infrasonic tremor
% K the a-priori chosen number of classes
options = [2.0 100 0.001 0]; K = 3;
[ctrs,U] = fcm(X,K,options);
maxU = max(U);
idx1 = find(U(1,:) =  = maxU);idx2 = find(U
(2,:) =  = maxU);
idx3 = find(U(3,:) =  = maxU);
plot(X(idx1,1),X(idx1,2),'go', X(idx2,1),X
(idx2,2),'b + ',…
X(idx3,1),X(idx3,2),'r*');
hold on
plot(ctrs(1,1),ctrs(1,2),'ko','MarkerSize',8,'linewidth',2)
plot(ctrs(2,1),ctrs(2,2),'ko','MarkerSize',8,'linewidth',2)
plot(ctrs(3,1),ctrs(3,2),'ko','MarkerSize',8,'linewidth',2)
xlabel('RMS of seismic tremor')
ylabel('RMS of infrasonic tremor')
title('FCM clustering of RMS seismic and infrasonic
tremor')
grid on
hold off
```

Obviously, in a multidimensional space μ_k and C_k are vectors and matrices, respectively. The parameters π_k can be regarded as the probability that z is produced by a random number generator with probability density $N(z|\mu_k, C_k)$. The parameters to be estimated are the number K of mixing components, the mixing parameters π_1, ..., π_K the mean vectors μ_k and the covariance matrices C_k. In order to estimate the model parameters, the iterative Expectation–Maximization (EM) algorithm (Xu and Jordan 1996) is considered. It is trivial to understand that this algorithm requires that a larger number of parameters with respect to the K-means needs to be estimated.

The application of the GMM approach to clusterize the RMS of seismic and infrasonic tremor is shown in Fig. 24.

The figure was obtained by using the Matlab code shown in Table 8.

The clustering results shown in Fig. 24 would apparently seem very different from those obtained with the same dataset, using the K-means and FCM algorithms. But this is only apparently true. Actually, using this algorithm requires the user to set a much richer number of parameters than the previous ones, starting from the structure of the covariance matrix that is not contemplated by K-means and FCM algorithms. Therefore the results strongly depend on these choices and the user is required make to a greater effort for a correct use of this algorithm.

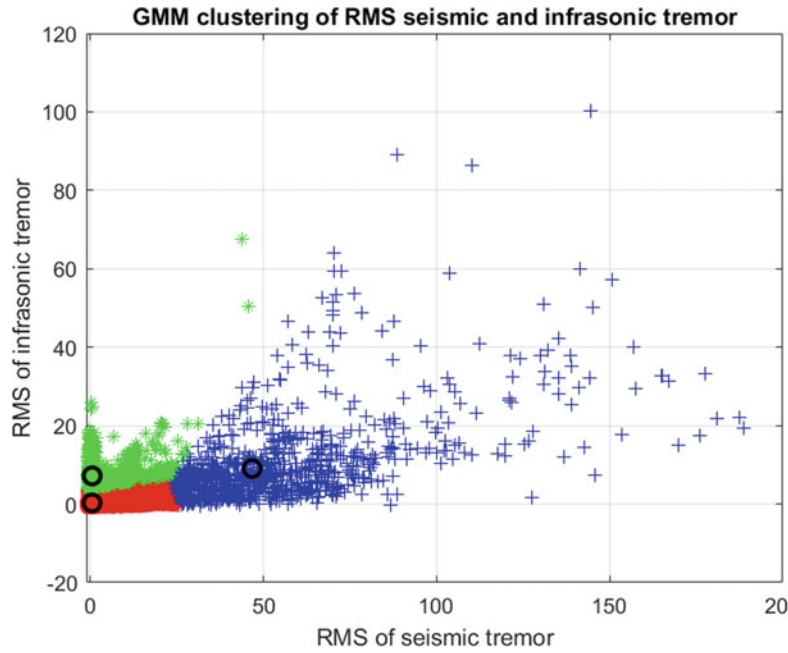


Fig. 24 Clustering the RMS of seismic and infrasonic tremor at Mt. Etna by using the GMM algorithm. Different colors indicate the three searched clusters. The circles symbols indicate the cluster centers

Table 8 Matlab code for using the Matlab *fitgmdist* function

```
%X is the matrix containing the samples seismic and
infrasonic tremor
% K the a-priori chosen number of classes
K = 3;
gmfit = fitgmdist(X,K,'CovarianceType','diag',…
'SharedCovariance',false,'RegularizationValue',0.01);
class = cluster(gmfit,X); % Cluster index
idx1 = find(class = = 1);
idx2 = find(class = = 2);
idx3 = find(class = = 3); ctrs=gmfit.mu;
plot(X(idx1,1),X(idx1,2),'ro', X(idx2,1),X
(idx2,2),'b + ',…
X(idx3,1),X(idx3,2),'g*');
hold on
plot(ctrs(1,1),ctrs(1,2),'ko','MarkerSize',8,'linewidth',2)
plot(ctrs(2,1),ctrs(2,2),'ko','MarkerSize',8,'linewidth',2)
plot(ctrs(3,1),ctrs(3,2),'ko','MarkerSize',8,'linewidth',2)
xlabel('RMS of seismic tremor')
ylabel('RMS of infrasonic tremor')
title('GMM clustering of RMS seismic and infrasonic
tremor')
grid on
hold off
```

7 Methods to Reduce the Dimensionality of a Dataset

In ML applications, it often happens to model physical processes starting from a dataset of measures z with high dimensionality. For example, suppose that we are considering seismic data recorded by a monitoring network having dozens of measurement stations and that want to understand the relationship between the intensity of seismic tremor and the categorical activity observed at the top of an active volcano. It is obvious that the seismic tremor time series recorded at different stations are strongly correlated and therefore considering vectors z with a number of components equal to the number of stations in the networks results in an unnecessary computational overload. Furthermore, operating with a dataset of high dimensionality poses serious problems for the graphical representation of the data. For these reasons it is a good

practice, before performing the modeling, to attempt a reduction of the dataset dimensionality. For this purpose, various techniques have been developed over the years, such as the so-called Principal Component Analysis (PCA), the Independent Component Analysis (ICA), the Self Organized Maps (SOM) and several others. For the sake of brevity a short description will be provided in this chapter only for the PCA and the SOM.

7.1 The Principal Component Analysis (PCA)

Given a dataset $z_i, i = 1, \cdots N_s$ of dimensionality N, the PCA consists in the search for a new dataset $\boldsymbol{y_i}, i = 1, \cdots N_s$ of dimensionality $D \ll N$, whose variables are referred to as the principal components. From the geometric point of view, this new set of variables represents a space in which each component $y_{i,j}$ is expressed as a linear combination of the $z_{i,j}$). In other terms, if z is the generic vector of the original dataset (referred to as the input) and $\boldsymbol{y}$ the corresponding vector in the lower dimensionality data set (the output dataset), then the idea is to look for a matrix W_D of dimension $D \times N$ which transforms z in y, by using a linear matrix equation of the form $y = W_D z$.

The search for these components will take place with the idea of maximizing the variance of the data in the new dataset. It can be shown, by linear algebra, that the search for the principal components corresponds to the search for the principal eigenvectors of a matrix. Indeed, indicating as C_Y and C_Z the covariance matrix of output and input dataset, which of course are represented by two matrices Y and Z, respectively, then the following equation hold

$$C_Y = W_D C_Z W_D^T = \Lambda_D \tag{57}$$

where Λ_D is a diagonal matrix of dimension D and W_D an orthogonal matrix (therefore invertible), i.e. the following equation holds: $W_D^T = W_D^{-1}$. The expression above for $D = N$ can be written as $C_Z W_N^T = W_N^T \Lambda_N$, where Λ_N is a diagonal matrix whose elements are the eigenvalues of C_Z and therefore the columns of W_N^T are the eigenvectors of C_Z. In summary, given a data set z of dimension N, computing the eigenvectors and the eigenvalues of the corresponding covariance matrix C_Z, we can build the matrix W_N^T ordering the eigenvectors as columns of C_Z, by decreasing eigenvalues. In such a way the matrix W_D, which is required for computing the reduced dimension data set, can be obtained from W_N^T by deleting the last $N - D$ rows, thus retaining the significant terms. For practical applications, the covariance matrix of the input dataset can be replaced by the sample covariance. However, it is to be stressed that despite the simplicity of this algorithm, the reduced dataset may not be suitable for classification purposes. Indeed, it is based on the idea of obtaining a reduced dataset which preserve as much as possible the variance and this is performed regardless of the class. In other terms, preserving the variance does not necessarily preserves the class distribution. This for instance occurred reducing the seismic tremor dataset considered for the application presented in Sect. 8 in Chapter "Machine Learning Applications in Volcanology and Seismology" (Nunnari 2021) of this book).

7.2 Self-organizing Maps

The Self-Organizing Maps (SOM) can be used both as a clustering approach or a technique for reducing the dimensionality of a given dataset. In some way, it can be considered a special version of the K-means algorithm, but in this case the cluster centers are constrained (i.e. mapped) to belong to a D dimensional space, usually $D \ll N$, where N is the measurement space dimension. To build such a low dimension grid the key idea is to preserve the topology of the input space. This simply means that the distance of objects in the input space must be as much as possible preserved in the mapped space. Thus, objects that belong in a neighborhood in the input space will belong to a neighborhood in the

mapped space. For this reason, this clustering approach is also used to reduce the dimensionality of a given measurement dataset. This goal is obtained by using a special kind of neural networks which make use of the so-called *winning rule*. Neurons of this kind of neural network are arranged into a grid of K points ($K = q_1 \times q_1 \cdots \times q_D$). The grid can have various topologies, such as for instance rectangular, hexagonal, random etc. In such a way various kinds of neighborhood criteria can be implemented. To each neuron a weighting vector $w_j, j = 1, \cdots, K$ is assigned in order to map the input space onto the output space. A training phase of the neural network will be performed by using the measuring vectors $z_n, n = 1, N_s$ of the training set. During the training phase of the neural network the weighting vector is updated according with the following algorithm:

1. Initialize the iteration index $i = 0$ and the weighting vectors $w_j^{(i)}, j = 1, \cdots, K$ assigned to the individual vectors $k_j, j = 1, \cdots, K$ of the grid. This can be done, for instance, assigning to each neuron a vector z_n randomly chosen from the training set.
2. For each measuring vector z_n of the training set, find the vector k of the grid which is closest to the vector z_n, i.e.which minimize $k(z_n) = \arg\min_{i\|} \| z_n - w_j^{(i)} \|$. The neuron of the grid corresponding to $k(z_n)$ is referred as the *winning neuron*.
3. Update the winning neuron and its neighbors by using the rule equation $w_j^{(i+1))} = w_j^{(i))} + \eta^{(i)} h^{(i)} \left(z_n - w_j^{(i)} \right)$, being $\eta^{(i)}$ the learning rate and $h^{(i)}$ and a suitable weighting function, which takes also into account the neighbors of the winning neuron.
4. Repeat steps 2 and 3 until the previous steps did not change significantly.

An example of application of this algorithm is given in Sect. 10 in Chapter "Machine Learning Applications in Volcanology and Seismology", dealing with a dataset of geophysical data recorded on Mt. Etna.

8 Software Tools for Machine Learning

In this short section we just want to mention some of the tools we have used for the purposes of this book. We are therefore aware that the list could grow considerably and that many other good ML tools have not even been mentioned. To the best of our knowledge, among the computer languages that are most used for implementing ML techniques we can mention Matlab (*The Mathworks Inc*®), Python and R. Furthermore, dedicated, open source libraries are available for each of them, and computer codes, also specialized for volcanic and geophysical data, can be found in open access repositories such as GitHub (Github 2020). In the Matlab framework we find as a suitable tool for ML the PRTool (Duin et al. 2017).

8.1 The MATLAB™ Statistical and Machine Learning Toolbox

The Statistics and Machine Learning Toolbox (The Mathworks 2017) provides functions and apps to describe, analyze, and model data. It allows descriptive statistics, visualizations, and clustering for exploratory data analysis. The toolbox provides supervised, semi-supervised and unsupervised machine learning algorithms, including support vector machines (SVMs), boosted decision trees, k-means, and other clustering methods. It is also possible to generate C/C + + code for embedded deployment. In the author's opinion, Matlab and in particular the Statistical and Machine Learning Toolbox (The Mathworks 2017) is one of the most powerful development environments, frequently updated and therefore enriched with the latest advances in the field of ML.

8.2 The Python Scikit-Learn Package

Scikit-learn is a free software (downloadable from https://scikit-learn.org/stable/) machine

learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. The Python packages for ML are so appreciated that it is not useless to spend words to advertise its use among users interested in ML applications. In addition to the richness of the functions developed, the user will also greatly appreciate the fact that it is *open source*.

8.3 The R Language

R is a programming language and free software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and ML practitioners for developing statistical software and data analysis. Similar to what we have declared for the Python environment, key points of R packages for ML are their pervasiveness and the open source aspect. It is possible to download R, from https://www.r-project.org/. To make more friendly the usage of R, it is possible to download an useful Integrated Development Environment (IDE), referred to as *RStudio* from https://www.rstudio.com/products/rstudio/download/.

8.4 The PRTools Library

PRTools is a library running in the MATLAB framework which supplies about 300 routines for traditional statistical pattern recognition tasks (see https://37steps.com/). It includes procedures for data generation, training classifiers, combining classifiers, features selection, linear and non-linear feature extraction, density estimation, cluster analysis, evaluation and visualization. One of the main advantages of this tool is the very high immediacy of implementation, which can therefore be useful for even less experienced users. But even the most experienced users will find a rich set of functions to develop their own applications.

References

Bishop CM (2006) Pattern recognitiom and machine learning. Springer, Berlin

Breiman L (2001) Random forests. Mach Learn 45:5–32

Cannavo F, Cannata A, Cassisi C, Di Grazia G, Montalto P, Prestifilippo M, Privitera E, Coltelli M, Gambino S (2017) A multivariate probabilistic graphical model for real?time volcano monitoring on Mount Etna. J Geophys Res Solid Earth 122:3480–3496

Cassisi C, Prestifilippo M, Cannata A, Montalto P, Patane D, Privitera E (2016) Probabilistic reasoning over seismic time series: volcano monitoring by hidden markov models at Mt. Etna, Pure Appl Geophys 173(7):2365–2386

Davies DL, Bouldin DW (1979) IEEE transactions on pattern analysis and machine intelligence PAMI-1

Dietterich TG, Bakiri G (1995) Solving multiclass learning problems via error correcting output codes. arXiv preprint cs/9501101

Duin RPW, Juszczak P, Paclik P, Pekalska E, De Ridder D, Tax DMJ, Verzakov S (2017) PRTools4 a matlab toolbox for pattern recognition

Escalera S, Pujol O, Radeva P (2010) On the decoding process in ternary error-correcting output codes. IEEE Trans Pattern Anal Mach Intell 32(7):120–134

Github (2020) GitHub. Retrieved from https://github.com/

Goodfellow I, Bengio Y, Courville A (2016) Deep learning, MIT Press. http://www.deeplearningbook.org

Xu L, Jordan MI (1996) On convergence properties of the EM algorithm for Gaussian mixtures. Neural Comput 8(1):129–151

Hajian A, Cannavo F, Greco F, Nunnari G (2019) Classification of mount Etna (Italy) volcanic activity by machine learning approaches. Ann Geophys 62 (2):231. https://doi.org/10.4401/ag-8049

Hastie T, Tibshirani R, Friedman J (2008) The elements of statistical learning-data mining, inference, and prediction. Springer, Berlin

Kong Q, Trugman DT, Ross ZE, Bianco MJ, Meade BJ, Gerstoft P (2019) Machine learning in seismology: turning data into insights. Seismol Res Lett 90(1), 3–14. https://doi.org/10.1785/0220180259

Lei B, Xu G, Feng M, Zou Y, Van der Heiden F, De Ridder D, Tax DMJ (2017) Classification, parameter estimation and state estimation—an engineering approach using MATLAB, Wiley

The Mathworks (2017) Statistical and machine learning toolbox User Guide

Nunnari G (2021) Clustering activity at Mt Etna based on volcanic tremor: a case study. Earth Sci Inform 14:11211143. https://doi.org/10.1007/s12145-021-00606-5

Parzen E (1962) On estimation of a probability density function and mode. Ann Math Statist 33(3):1065–1076. https://doi.org/10.1214/aoms/1177704472. https://projecteuclid.org/euclid.aoms/1177704472

Rouseeuw PJ (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20(1):53–65

Xu D, Tian D (2015) A comprehensive survey of clustering algorithms

Van Rossum G, Drake FL (2009) Python 3 reference manual. CreateSpace, Scotts Valley, CA

Machine Learning Applications in Volcanology and Seismology

Abstract

The purpose of this chapter is to show some applications in volcanology and seismology of Machine Learning methods as reported in literature.

1 Introduction

Volcanologists consider a large variety of techniques in order to understand activity inside a volcano. Similarly, seismologists do in order to estimate the seismic hazard for a given area. For monitoring purposes, several kinds of signals are considered, such as seismic, magnetic, electromagnetic, ground deformation, infrasonic, thermal and geochemical data. These signals, possible in a multi-parametric approach, are analysed with the aim of estimating the probability of an unrest of the volcanic activity. Before to explain how ML techniques are considered to help in reaching this goal, it is better to clearly state that forecast volcanic eruptions, specifying when, where and how an eruption will occur, is even hard and at the present there are no serious evidences that this goal can be reached in the near future. A similar conclusion holds also for the earthquake prediction. Nevertheless, many volcanoes and seismic areas are monitored by institutional organizations that try to estimate at least the probability of the different hazardous volcanic and seismic events. Seismic data are always the fundamental of any monitoring system, and in particular the analysis of continuous volcanic tremor has in fact a great potential. Indeed, many processes in and around volcanoes can generate earthquakes. Most of the time, these processes are faulting and fracturing that does not lead to an eruption. However, volcanic earthquakes do occur as magma and volcanic gases rise to the surface from depth, which involves significant stress changes in the crust as the material migrates upward. This explains why the large part of ML applications proposed in literature from a long time ago, refer with seismic signals and tremor.

2 ML to Classify Seismic Data

In order to automate the work of classifying large amounts of seismic data, several authors have proposed the application of ML algorithms. Indeed, while the detection of seismic events is commonly considered a problem that is nowadays efficiently solved, the classification of events is still performed by human experts. Volcano seismologists classify several types of seismic events to better understand how magma and gases move towards the surface. Classes of seismic waveforms have been recognized in volcanic areas such as:

- Volcano-tectonic (VT) earthquakes, which represent brittle failure of rock, the same

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

A. Hajian et al., *Intelligent Methods with Applications in Volcanology and Seismology*, Advances in Volcanology, https://doi.org/10.1007/978-3-031-15432-4_3

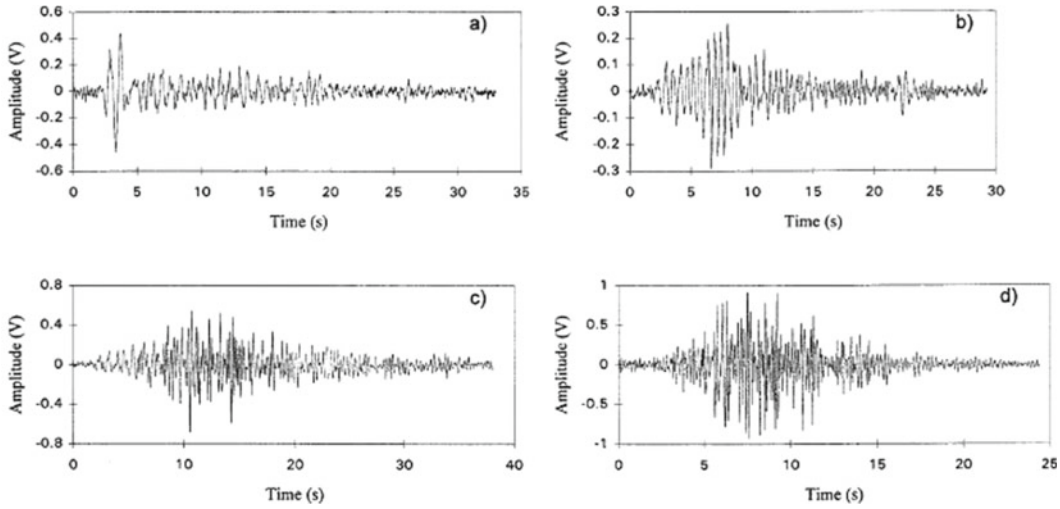


Fig. 1 Four classes of waveforms observed at Stromboli volcano, labeled as K, L, M and N, respectively (reprinted from Falsaperla et al. 1996)

process that occurs along purely tectonic faults;
- Long-period (LP) or low-frequency (LF) earthquakes, which are attributed to expansion and compression of sub-horizontal cracks filled with steam or other ash-laden gas;
- Explosion quakes, which are events observed in volcanic areas such as Stromboli.
- Tremor (TR) events, which are characterized by presenting a duration longer than LP.

The above classification is not exhaustive and other classes could be added as testified by a vast literature. Not all authors aim to discriminate against all classes of events listed above. Indeed, most of them aim to recognize two or three classes that they consider prominent for their purposes. A feature shared by all authors is that of not directly using the waveforms of seismic signals for classification, but extracting features before performing the classification. This often allows to decrease the dimensionality of the problem, thus improving the classifier performances. Indeed, it is known that the performance of a classifier quickly falls as the size of the feature space increases.

As an example, in Falsaperla et al. (1996), with the aim of classifying events recorded in the Stromboli area into four classes (see Fig. 1), it was proposed a strategy based on representing the original waveforms by using a combination of their autocorrelation function plus a feature, referred to as the envelope function, which accounts for the shape of waveforms.

Since it is almost impossible to mention, even partially, all the features that have been proposed for the classification of seismic waveforms, in this book, in order to give the readers an idea, we will refer to the list proposed in the recent work by Falcin (2021), who compute a total number of 34 features, distinguished into 9 statistical features, 9 entropy-based features and 16 shape descriptors. Furthermore, these 34 features are computed into 3 different representations of the waveforms, namely the time domain, the frequency domain and the so-called cepstral domain. The cepstral domain describes the periodic properties of the signal, as commonly used in speech processing, and it is obtained by computing the Fourier transform of the logarithm of the signal spectrum. The 9 statistical considered features are the following: the length of the waveform, the mean, the standard deviation, the skewness, the kurtosis, the index of central energy, the RMS bandwidth, the mean skewness and, finally, the mean kurtosis. Among the

entropy features, the Shannon entropy and the Renyi entropy are considered, while among the shape descriptors the rate of attack and the rate of decay are taken into account.

As concerning the ML approaches proposed for classification of seismic events we can mention Decision Tree (Allen 1978), Multi-Layer Perceptron Neural networks (Falsaperla et al. 1996), Fuzzy logic and decision rules (Hibert et al. 2014), Hidden Markov Models (Bentez et al. 2006), Logistic Regression and SVM (Langet et al. 2014), Random Forest (RF) and SVM (Malfante et al. 2018a, b), Deep Neural networks (Titos et al. 2018), spectrogram cross-correlations in conjunction with the K-nearest neighbors algorithm (Curilem et al. 2018).

Although this list of methods is far from exhaustive, the classification of seismic waveforms follows a common pattern, schematically shown in Fig. 2, which consists of the following steps.

- Preparation of a data set: waveforms are selected and usually those characterized by a low signal to Noise ratio (SNR) are discarded.
- Labels are associated with each waveform by human experts.
- Features are extracted from the original waveforms.
- The data set is divided in at least two sets considered respectively for training and testing the classifier. Alternatively, the k-fold cross validation strategy can be adopted.
- After training the classifier performance is assessed, usually in terms of Accuracy, Precision, Overall Accuracy, etc. (see Sect. 5 in Chapter "Machine Learning: The Concepts" for the description of these performance indices).

As regards the results reported in the literature, these obviously depend on many factors such as the area where the seismic events were recorded, the number of classes to be discriminated against, the method used for the training of the classifier etc. In principle, expressing the performance in terms of the overall accuracy, it normally ranges from 70% to 90%. Therefore, it can be affirmed that for this kind of application the ML algorithms proposed in literature can be considered reliable enough.

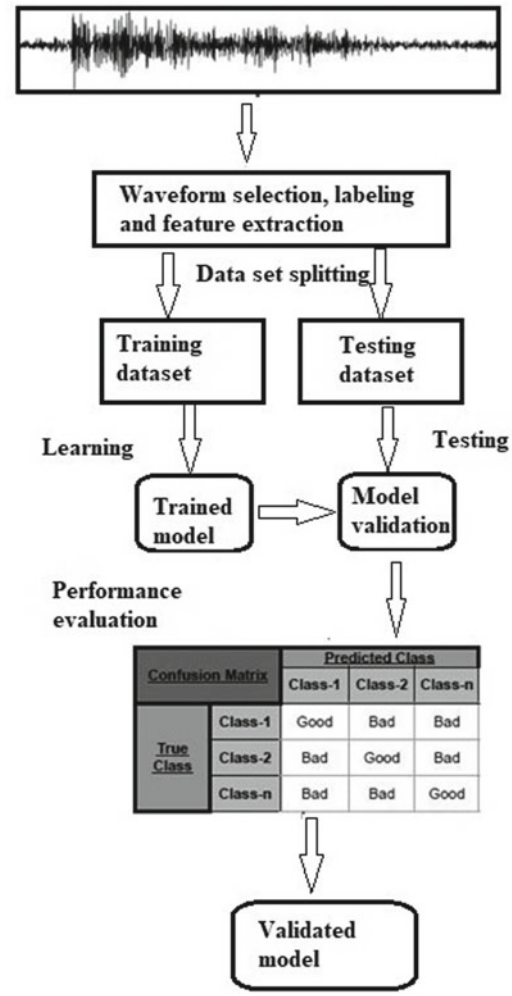


Fig. 2 Schematic of the procedure for seismic waveform classification

3 Hidden Markov Model to Classify Volcanic Activity

In this section we summarize the application of Hidden Markov Models (see Sect. 4.11 in Chapter "Machine Learning: The Concepts") to classify the volcanic activity at Mt Etna, by using the RMS of volcanic tremor recorded at 16 stations, as described by Cassisi (2016). The state of the volcano was represented by a vector consisting of the following categorical variables:

$$S = \{QUIET,\ PRE - FOUNTAIN,\ FOUNTAIN,\ POST - FOUNTAIN\}$$

The seismic tremor was considered to generate a finite set of observation symbols. In order to transform the real valued tremor time series to categorical, a technique known in literature as the Symbolic Aggregate approXimation (SAX) (Lin et al. 2007) approach was considered, as schematically shown in Fig. 3. Real-valued signals (in the example a frame of 128 samples) was first represented by using the Piecewise Aggregate Approximation (PAA) algorithm, which consists of computing the mean value in windows of length w = 8 samples. After this, the PAA frames were translated into equiprobable symbols of the set $\{a, b, c, d, e, f, g, h\}$. However, after some trial the authors decided to double the length of the alphabet which become:

$$O = \{a-, a+, b-, b+, c-, c+, d-, d+, e-, e+, f-, f+, g-, g+, h-, h+\}$$

In this new version the sign + was added to each symbol, if the present signal was greater than the previous on the time series; otherwise a sign − was added. Probably, this was motivated to take into account the sign of the first derivative of the signal. Moreover, they added the symbol _ (underscore) to represent the absence of signal. Therefore, the final alphabet for representing the RMS of seismic signals consists of 17 symbols. Furthermore, the set of states S, was also further extended by adding two more states, referred to as *NO-SIGNAL* and *EARTHQUAKE*, in order to take into account the absence of signal in a given station, or that an earthquake occurred. A HMM model was trained by using six months of tremor (from 1 January 2011 to 1 July 2011), for each of the 16 seismic stations. For each seismic station a custom emission matrix was computed by using the Baum-Welch algorithm. Differently from the emission matrices, all stations shared the same transition matrix. This choice was motivated considering that the volcano state does not depend on the observations of individual stations but, rather, the stations record a shared phenomenon. The popular Viterbi algorithm was considered for decoding the observed sequences of the tremor signal into the states of the volcano. Due to the fact that each station is decoded individually, the final decision about the status of the volcano, at each time step, was performed by a voting mechanism among the stations, consisting of labelling the state of the volcano as the maximum occurrence from the 16 stations of the seismic network.

The obtained HMM was simulated for the time period starting from 1 July 2011 to 30 June

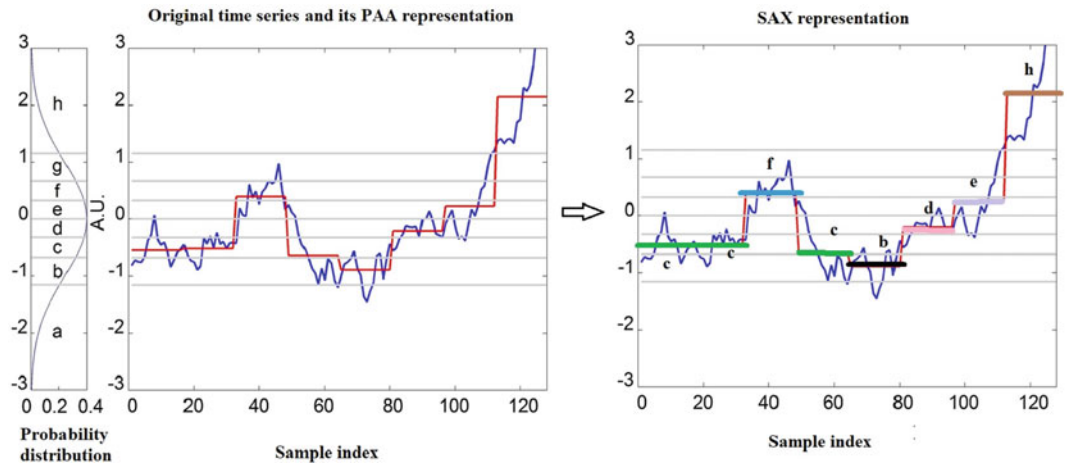


Fig. 3 Schematic of the Symbolic Aggregate approXimation (SAX) (Cassisi 2016)

2013. The authors claim that, with a time step of 5 min, during the testing period, the HMM model was able to find 45 sequences longer than 15 min, related to the FOUNTAIN state. For each of these sequences it was checked if they correspond to actual lava fountain episodes. The authors claim that 34 lava fountains that really occurring at the NSEC crater of Mt Etna were detected with a TPR = 100% and a FPR = 24.5%. The authors motivated the relatively high rate of FPR due to Strombolian activity occurring in different summit craters (the NSEC and BN craters) with respect to the considered one. In spite of these, there were also cases of non-paroxysmal explosive activity that were not identified by the model, among which the most important is the Strombolian activity that occurred at the BN crater in July 2011.

4 Earthquake Detection and Phase Picking

Automated detection and picking of earthquakes is a basic problem in seismology. Several trigger algorithms are known and successfully used nowadays, ranging from the very simple amplitude threshold type to the more sophisticated ML methods. Among the amplitude threshold approaches, the Short-Time Average (STA)/Long-Time-Average (LTA) is the most broadly used algorithm in weak-motion seismology (Allen 1978). It continuously calculates the average values of the absolute amplitude of a seismic signal in two consecutive moving-time windows. The short time window (STA) is sensitive to seismic events while the long-time window (LTA) accounts for the temporal amplitude of seismic noise at the recording site. After computing the STA/LTA ratio, it is possible to establish, based on a threshold criterion, the triggering on and off time instants, as schematically shown in Fig. 4.

Despite the effectiveness of simple algorithms such as STA/LTA a large number of ML approaches have been recently proposed, in particular based on the use of Convolutional Neural Networks (CNN). As an example, the use of a CNN to perform the phase detection, as described by Ross et al. (2018) and reprinted by (Kong et al. 2019) is shown in Fig. 5.

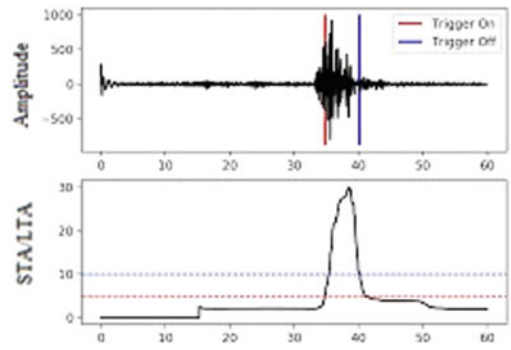


Fig. 4 Schematic of the threshold criterion, the triggering on and off time instants for STA/LTA

It is possible to see a CNN, supplied with the seismic waveforms recorded in the N, E and Z channels. Seismic features are then extracted and processed by a fully connected neural network which returns, at each time step, the probability of detection of P and S phases, or the probability that the recorded waveform must be considered as a Noise. One of the advantages of using this approach is that, after training, the CNN is able to extract the features and provide results directly from row data very quickly.

5 Earthquake and Early Warning

Earthquake Early Warning (EEW) Systems are based on the idea of providing information about an incoming earthquake, within the first few seconds of the onset of the P-wave, i.e., before the damaging S-wave arrives. Indeed, the S wave travels at a slower velocity than the P-wave, and therefore the travel time difference can be useful for issuing an alert. Nakamura (1988) was the first who proposed UrEDAS (Urgent Earthquake Detection and Alarm System) an EEW system for earthquake detection, clearly outlining the qualities that should characterize a similar plant, which should be fully automated, quick and reliable, small and chip, independent of other systems and easy to connect to communication networks.

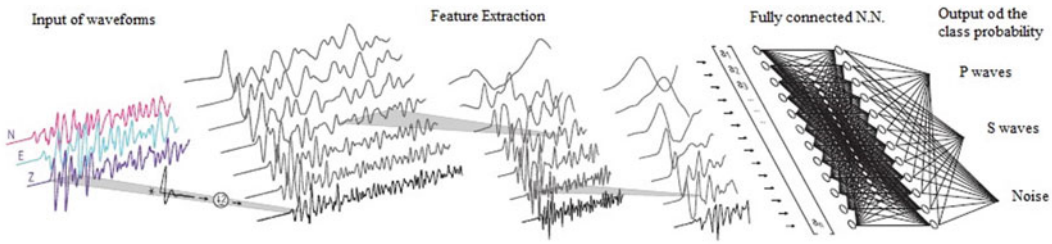


Fig. 5 Schematic of the seismic phase detection by using a DNN (Kong et al. 2019)

Estimation of earthquake early magnitude at various regions can be performed by different approaches such as Zero-crossing (Allen and Kanamori 2003), Wavelet multi-scale (Simons et al. 2006), Short Time Fourier Transform (STFT) (Wurman et al. 2007), Zero Crossing and peak ground displacement (Tsang et al. 2007), Peak ground acceleration (Lin and Wu 2010). Since the aim of this section is to illustrate the role of SVM, we describe in this chapter an application due to Reddy and Nair (2013), based on the joint use of Wavelet multiscale and SVM. In order to implement the system, a seismic dataset was split into two equal parts, of low (M_l) ($M < 5.2$) and high (M_h) magnitudes. By using an approach based on the C_7 wavelet coefficient obtained by representing the seismic waves in terms of Wavelets. In particular, Reddy and Nair (2013) estimated the following linear models:

$$M_l = 1.25log(C_7) + 1.8 \quad (1)$$

$$M_h = 1.41log(C_7) + 1.7 \quad (2)$$

This achievement was then further improved by using a ML approach, based on SVM to search for best fit regression lines, between the average of the classified threshold coefficients at a scale C_7 and the local earthquake magnitude, as schematically shown in Fig. 6. Thus, after the search by SVM, the authors were able to update the models as:

$$M_l = 1.07log(C_7) + 1.9 \quad (3)$$

$$M_h = 2.40log(C_7) - 2.2 \quad (4)$$

claiming, after computing the average error, that they perform better than the previous ones.

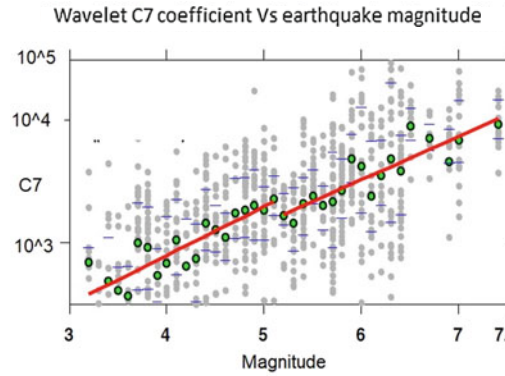


Fig. 6 Absolute magnitude of the first significant classified wavelet coefficients at scale 7 obtained after the SVM approach (Reddy and Nair 2013)

6 Ground Motion Prediction

Ground-motion prediction is a topic related to earthquake hazard assessment. The simplest way to represent the relation between the log of the ground motion Y and the seismic features, is a linear regression model as expressed by (5):

$$\ln(Y) = \beta_0 + \beta_1 M + \beta_2 \ln(R) + \varepsilon;\ \ \varepsilon \sim \eta\left(0, \sigma^2\right) \quad (5)$$

where M is the earthquake magnitude, R the source-site distance and ε a random Gaussian distributed variable with zero mean and variance σ^2. The ground motion variable Y can assume different meanings. It is usually representing the Peak Ground Acceleration (PGA), but it can also represent other variables such as the Peak Ground Velocity. However, in the ground motion problem, the physical phenomena are

intrinsically non-linear and thus the linear model (5) may be, in some cases, not reliable enough. To overcome these shortcomings, the model can be generalized as:

$$\ln(Y) = f(X, \theta) + \varepsilon \tag{6}$$

where X represents the vector of features, θ the model parameter vector and ε the random variable mentioned above. In order to estimate the unknown non-linear function f, several ML approaches can be used. For instance, advantages and disadvantages of using Artificial Neural Networks, Random Forest or SVM to solve the described problem were studied by Khosravikia and Clayton (2021). This study was performed on a database consisting of 374 different earthquakes recorded at 209 seismic stations in Texas, Oklahoma, and Kansas since 2005, which has been interested in increased seismicity rate due to natural gas production and petroleum activities.

Performances of ANN, RF and SVM based approaches, in comparison with linear regression models, were assessed by computing the regression coefficient R in the plane Predicted versus Measured *ln(PGA)*. Obtained values for the R coefficient by using the mentioned approaches were: $R = 0.921$ for the ANN, $R = 0.958$ for the RF, $R = 0.936$ for the SVM and $R = 0.898$ for the linear model. On this basis the authors conclude that when a sufficient dataset is available, all the considered ML algorithms tend to provide more accurate estimates compared to the conventional linear regression-based method, and in particular, the Random Forest outperforms the other algorithms. However, the authors still recommend the conventional linear method, when limited data is available.

7 ML for Volcanic Activity Monitoring Based on Images

Techniques for systematic monitoring of active volcanoes using images which can be recorded both by satellite techniques or thermal cameras installed at ground level, usually arranged to form networks, have been proposed in literature. A prototype of this kind of systems referred to as VAMOS (**V**olcanic **A**ctivity **Mo**nitoring **S**ystem) was designed for monitoring volcanic activity at Mt Etna and Stromboli volcanoes (Bertucco et al. 1999). The architecture of the VAMOS system is shown in Fig. 7. It was able to acquire images by a thermal camera appropriately installed in order to monitor the summit crater area, process them in real time and establish if events of volcanological interest, such as an explosion, lava fountain, occurred. Nowadays this prototype system has been replaced by a modern one which at Mt Etna consists of five fixed continuously operating thermal cameras located around the summit area. Images are transmitted at the INGV central room and displayed in real-time to allow continuous monitoring of the volcano. In addition to the hardware architecture of the system, the software has also evolved and currently the event recognition system is based on ML algorithms, as described by Corradino et al. (2020), who proposed to classify volcanic activity from images into five classes, labeled as:

1. no activity, characterized by low degassing/ash level, cooled lava flow and deposit;
2. degassing, characterized by lithic ash emission or gas plume without magmatic explosions;
3. effusive, characterized by effusive activity/incandescent deposits;
4. explosive, characterized by Strombolian activity with or without plume, lava fountain, with or without plume, and explosive activity in general;
5. explosive and effusive.

In the mentioned paper, classification was performed in two steps: in the first step images were classified into two classes, referred to as clear (meaning sunny day, clear night, slightly foggy and slightly cloudy scenes), and fuzzy (meaning heavily foggy and cloudy scenes). To perform this step a Decision Tree (DT) classifier was considered. During the training phase, appropriately selected sets of features with their

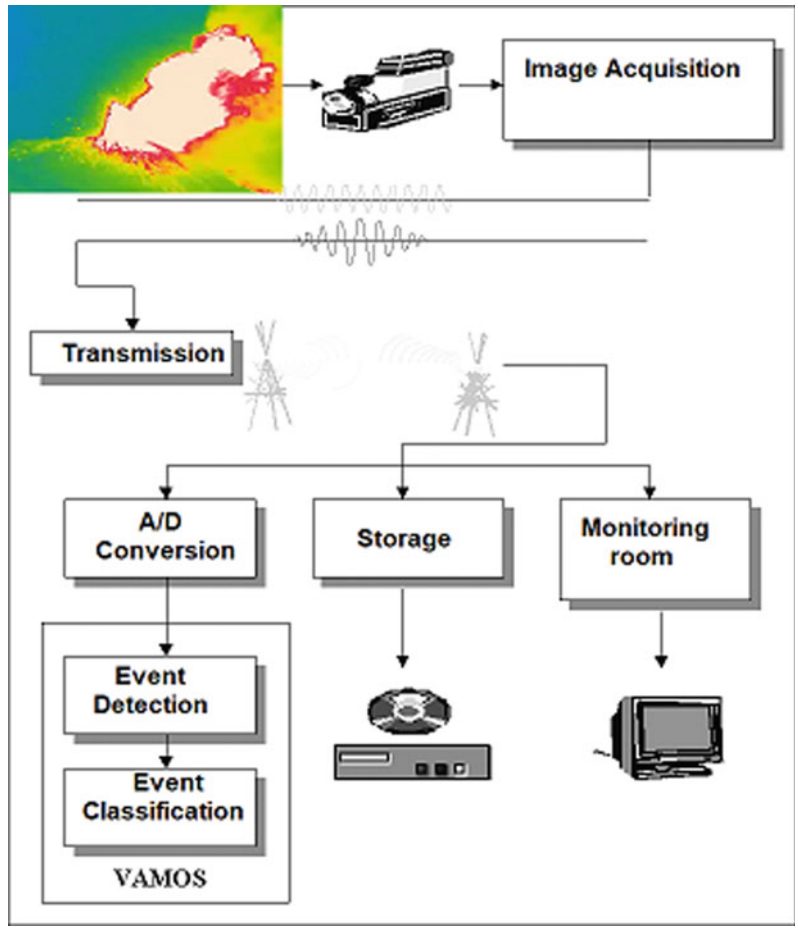


Fig. 7 Structure of VAMOS system for volcano activity monitoring based on images recorded at ground level by using thermal cameras (Bertucco et al. 1999)

corresponding classes are provided to the decision tree to learn the node criteria and build up the DT. To discriminate between clear and fuzzy grey color histograms were considered. To perform this step, the authors found that a 10-level grey colour histogram provides a good representation of the thermal image, allowing the DT classifier to derive appropriate thresholds for each intensity level. Thus, a vector was built for each image and the corresponding class was assigned following this rule: 0 if clear and 1 if fuzzy.

The authors claims that DT classifier is able solve the two class problem (clear and fuzzy) with high accuracy, as shown in Table 1.

Each value of the matrix represents the number of observations that are correctly and incorrectly classified (redrawn after Corradino et al. 2020).

In the second step, images recognized as *clear* class were further classified into the five classes described above, by using a Bag of Word (BoW) classifier, i.e. a special kind of classifier, which considers image features as words.

The performances of the BoW obtained by Corradino et al. (2020) for classifying clear

Table 1 Confusion matrix for the output of the DT classifier

True	Predicted	
	Clear	Fuzzy
Clear	0.996	0.004
Fuzzy	0.003	0.997

Table 2 Confusion matrix of the Bag of Word based classifier (redrawn after Corradino et al. 2020)

True	Predicted				
	Explosive	Effusive	Explosive and effusive	Degassing/ash emission	No activity
Explosive	0.960	0.009	0.0046	0.027	0.000
Effusive	0.007	0.977	0.002	0.014	0.000
Explosive and effusive	0.008	0.004	0.984	0.004	0.000
Degassing/ash emission	0.000	0.000	0.000	0.620	0.380
No activity	0.020	0.000	0.010	0.000	0.970

images into the five above mentioned volcanic activity classes are reported in Table 2.

On the basis of above results the authors concluded that the considered ML approach was able to classify with high accuracy four of the five classes (i.e. explosive, effusive, explosive and effusive and no activity) and with lower accuracy the class referred to as degassing/ash emission.

8 Multi-parametric Approaches to Classify the Volcanic Activity

A multi-parametric approach to classify the volcanic activity at Mt Etna into three classes, referred to as Q (Quite), S (Strombolian) and P (Paroxysm), based on geophysical time series was firstly studied by Cannavò et al. (2017), who implemented a Bayesian Network (BN) Graphical model. The dataset taken into account for this work was recorded on Mt Etna from 2011 to 2015 with a sampling time of 10 min and consists of the time series shown in Fig. 8.

The six time series considered were the following:

1. The activity observed at the summit of the volcano. It is a categorical time series $G = \{0, 1, 2\}$. The symbols 0, 1, 2 indicate Quiet, Strombolian and Paroxysm activity.
2. The weighted normalized median of seismic tremor (RMS). This feature was obtained by averaging, over 16 stations, the vertical component of the seismic signal, tacking 10 min long time windows.
3. The normalized volcanic tremor source depth obtained by a grid search method, based on a spatial seismic amplitude distribution and assuming the propagation in a homogeneous medium, within 30 min long time windows. The use of relatively long-time windows was necessary to reduce the effects of transients on the volcanic tremor locations.
4. The normalized number of infrasonic events, obtained by a large number of infrasonic events, recorded by the permanent infrasonic network during the considered time interval (2011–2015).
5. The feature normalized radar RMS, obtained by processing the total radar echo backscattered by particles in the atmosphere crossing the beam of a Doppler radar, referred to as VOLDORAD 2B, installed in the upper southern flank of the volcano (2600 m a.s.l.).
6. The normalized tilt derivative obtained by using the data recorded by the tiltmeter station referred to as CDB.

In Hajian et al. (2019) by using the above mentioned dataset two kinds of classifiers were trained: a Decision Tree (DT) and a K-Nearest Neighbour (KNN). Results obtained in terms of TPR and the TNR for each class and for each kind of classifiers are reported in Fig. 9, which shows that the fine DT classifier slightly

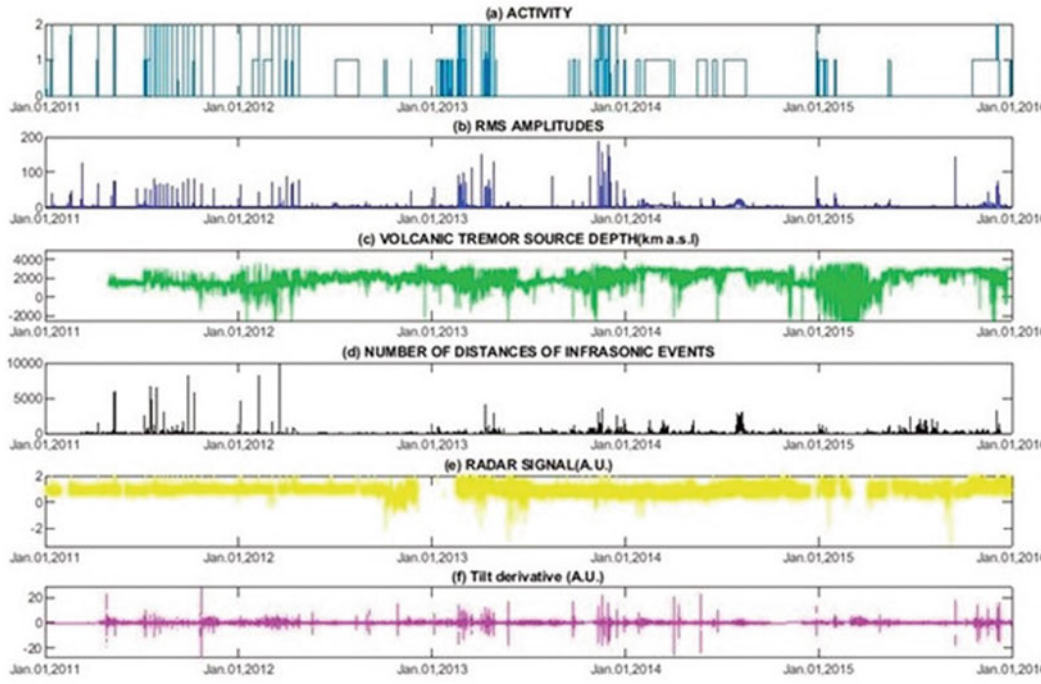


Fig. 8 Mt Etna **a** activity, **b** RMS amplitudes, **c** volcanic tremor depth, **d** number of distances of infrasonic events, **e** radar signal, **f** tilt derivative, from 2011 to 2015 (Hajian et al. 2019)

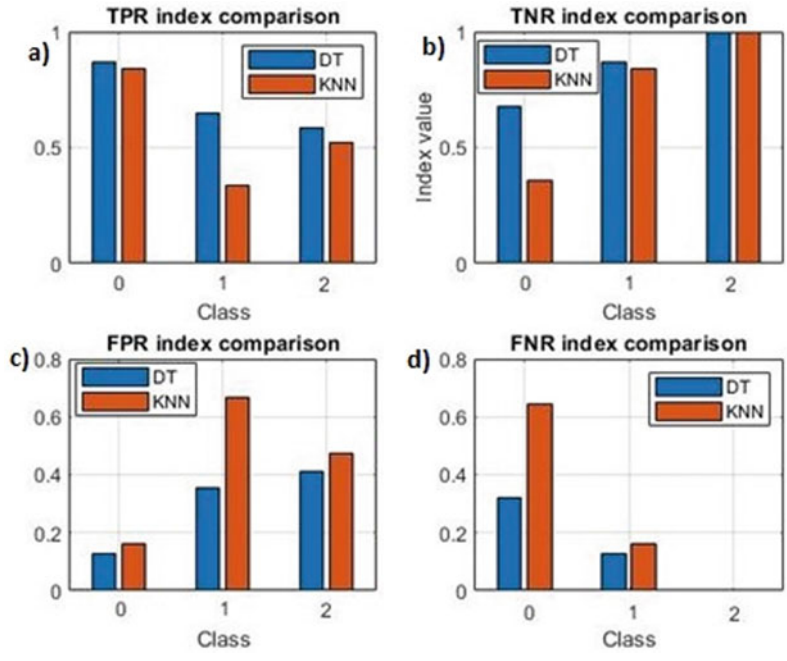


Fig. 9 A direct comparison between the DT and KNN classifiers in terms of TPR, TNR, FPR and FNR (Hajian et al. 2019)

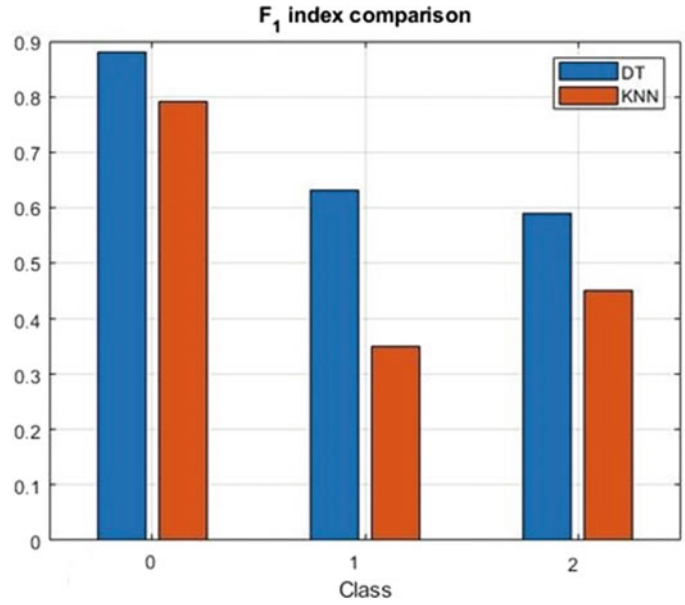


Fig. 10 A direct comparison between DT and KNN in terms of the f_1 index (Hajian et al. 2019)

outperforms the fine KNN for all the classes and indices. This result was also confirmed by computing the f_1 index, as shown in Fig. 10.

Another application of supervised learning to classify volcanic activity at Mt Etna was performed by Nunnari (2021) considering the same dataset as in Hajian et al. (2019). However, in Nunnari (2021) after realizing that among the five features considered in Hajian et al. (2019), the seismic volcanic tremor is the most important one, it was proposed to perform the volcanic activity classification on the basis of the tremor feature only. For this application, the following classification approaches were considered:

- A Fisher Discriminant model, simply referred to as DISC;
- a Multiclass error-correcting output model, referred as ECOC;
- an Ensemble model, referred to as ENSE;
- a K-Nearest Neighbor model, referred to as KNN;
- a Naive-Bayes model, referred to as NBYE;
- a Decision Tree for multiclass classification, referred to as TREE.

In Nunnari (2021), the seismic tremor recorded from 2011 to 2015 by the whole Etna seismic network, whose topographic map is shown in Fig. 11, was initially considered.

However, following the idea that the time series of seismic networks are necessarily highly correlated, since generated by the same volcanic source, the Principal Component Analysis (PCA) was performed.

The cumulative variability explained, tacking as input the original 17-D dataset, versus the number of principal components, is shown in Fig. 12.

It is possible to see that the first two or three principal components are able to account for 96% and 99% of the cumulative variability, respectively. Therefore, the PCA analysis points out that the original 17-D dataset could be reduced to 2-D or 3-D. However, bearing in mind that the principal components of the reduced dataset not necessarily preserve the class (see Sect. 7.1 in Chapter "Machine Learning: The Concepts"), for the considered application, in addition to the reduced dataset obtained through the PCA, it was taken into account another reduced dataset obtained by heuristically choosing two stations of the network. Furthermore, it was considered the option of reducing the 17-D dataset to 1-D, by simply averaging the seismic tremor among all the stations in the network through expression (7).

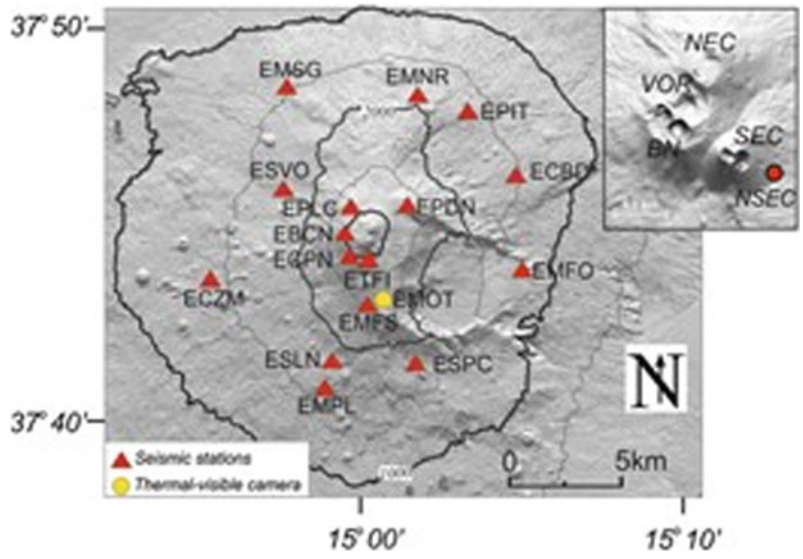


Fig. 11 Map of the seismic network at Mt Etna (Nunnari 2021)

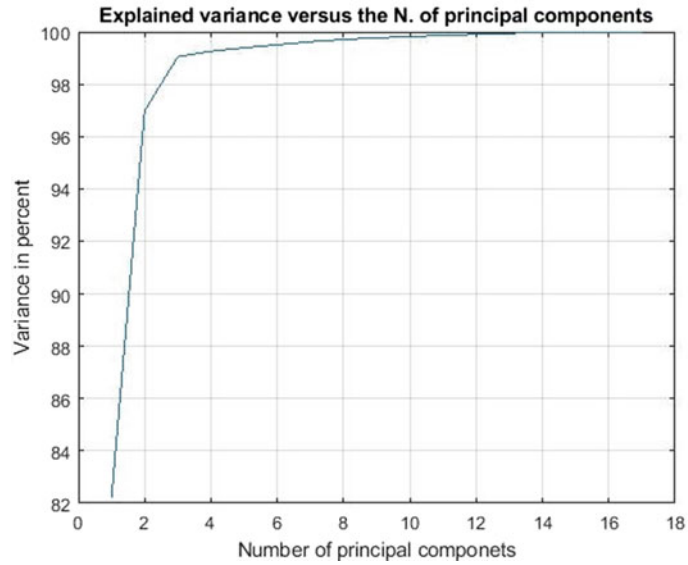


Fig. 12 Cumulative explained variance versus the principal components (Nunnari 2021)

$$T_{RMS}(t) = \frac{1}{n}\sum_{i=1}^{n}\frac{RMS_i(t) - \mu_t\{RMS_i(t)\}}{\sigma_t\{RMS_i(t)\}} \quad (7)$$

In this expression, learned from (Cannavò et al. 2017), the index, i refers to the n seismic stations, μ_t represents the median operator over two years previous the time t, and σ_t represents the interquartile range over the same period.

Summarizing, the following three reduced data set were considered to perform the supervised classification:

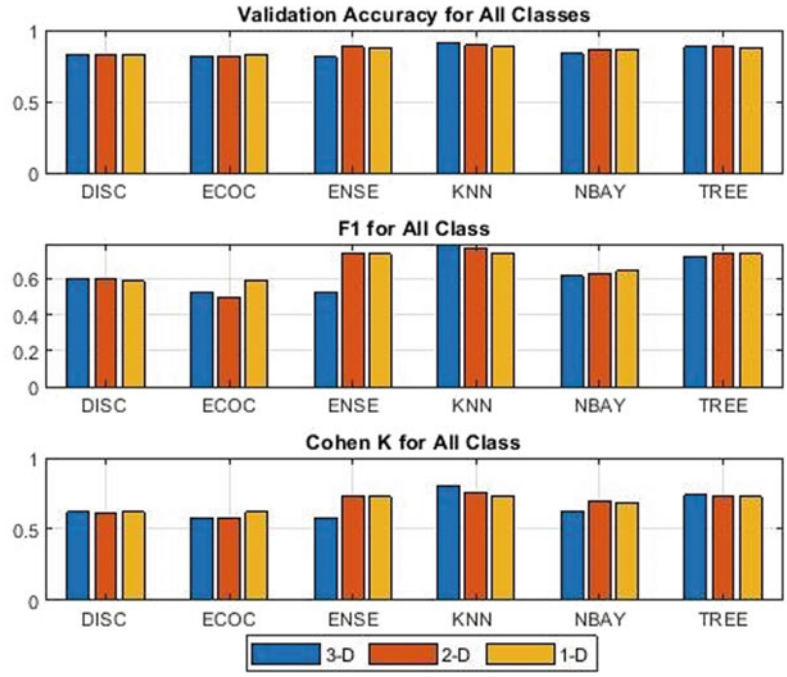


Fig. 13 Global indices for the three options (Nunnari 2021)

1. The 3-D dataset represented by the first three PCA components;
2. a 2-D dataset consisting of tremor recorded at two selected stations of the network, namely ESLN and EBCN;
3. the 1-D dataset obtained averaging the RMS tremor over the whole network, by using expression (7).

To obtain the 2-D dataset, the stations ESLN and EBCN shown in Fig. 11, were selected, according to the criteria that are located far from the summit crater area and therefore less sensitive to noise.

Numerical results obtained, in terms of global indices, i.e. computed averaging over the classes, namely the Accuracy, the f_1 index and Cohen's K, are shown in Fig. 13. Roughly speaking, it was possible to say that in terms of *Accuracy*, there are not significant differences between the various classifiers and to the use of a particular kind of reduced dataset. Instead, the global f_1 index points out a slight advantage using the ENSE and the KNN.

9 Unsupervised Classification of Volcanic Activity

Clustering of volcanic activity is delegated to human experts, as it requires a timely consultation of various kinds of reports and instrumental data. This activity is then all the more demanding as the time interval between samples is smaller. With the aim of contributing to automate this activity (Nunnari 2021) investigated the unsupervised classification of volcanic tremor into three classes which should resemble the Q, S and P labels used by the expert classification shown in Fig. 8 and also taken into account in Sect. 8.

Before to further proceed with results claimed by Nunnari (2021), it is worth nothing that unsupervised classification of seismic tremor has been previously addressed by others authors, and in particular we mention the work made by Langer et al. (2011), who applied a combination of Self-Organized Maps (SOM) and Fuzzy clustering, with the aim of detecting imminent eruptive activity at Mt Etna. Furthermore, we

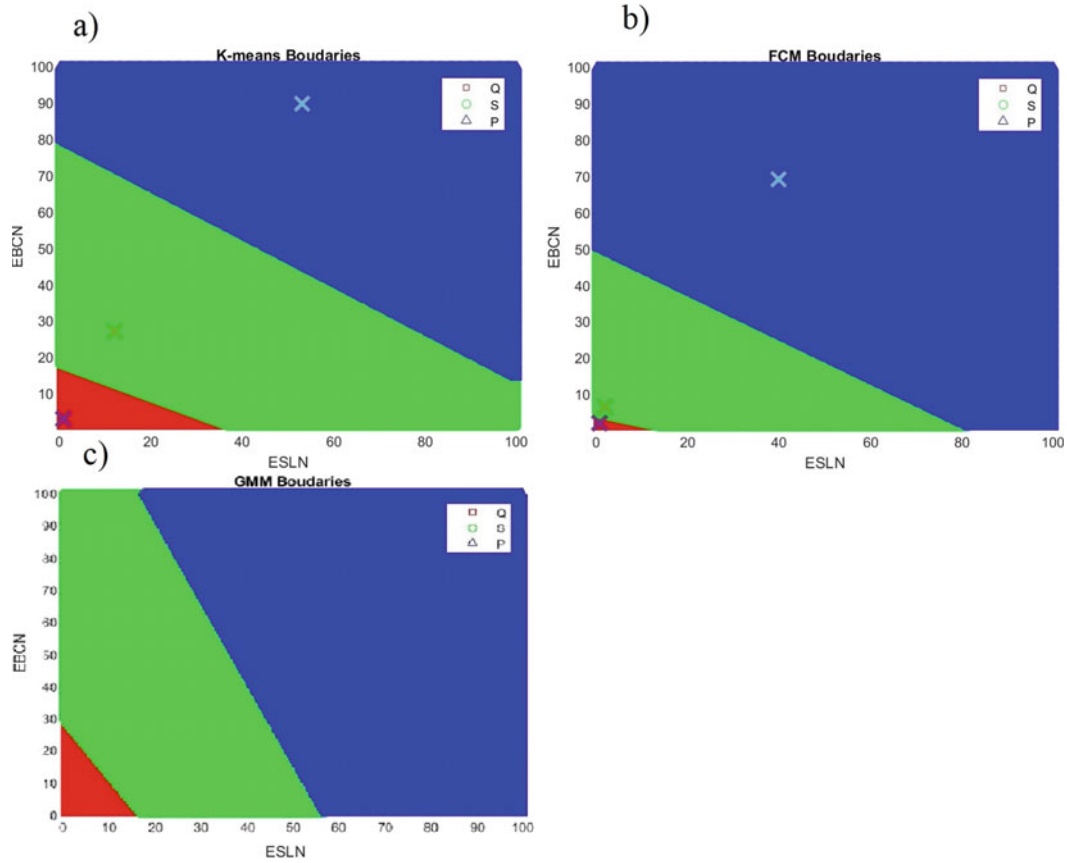


Fig. 14 Classification regions obtained performing: **a** the K-means (cluster centres for the three classes are also indicated by the 'x' symbol), **b** the FCM algorithm, **c** the GMM approach (Nunnari 2021)

mention that these and other ML approaches have been the basis for a recent volume by Langer et al. (2020) which address advantages and pitfalls of pattern recognition in selected geophysics applications.

In Nunnari (2021), in order to perform the clustering, three popular approaches, namely the K-means, the Fuzzy c-means and the Gaussian Mixture Models (GMM) were considered. Although the activity of an unsupervised classifier, once trained, can be carried out indifferently on-line and off-line, the purpose that the author was not that of monitoring volcanic activity but to assess the accuracy off-line. For this purpose, the time series of the classes produced by the various non-supervised classifiers have been compared with the true time series of the classes provided by expert volcanologists and used in the previous Sect. 4 in Chapter "Machine Learning: The Concepts" for the training of various supervised classifiers. Assessing of the unsupervised classifier was performed by evaluating the f_1-score, the Accuracy and the Cohen-K indices (see Sect. 5 in Chapter "Machine Learning: The Concepts"). For the sake of brevity, here we will not enter into questions such as if it is better to cluster the seismic tremor averaged over the whole network or to consider a subset of them. We have addressed these questions in the previous Sect. 4, dealing with the use of supervised classifiers. Therefore, now we simply refer to results that were achieved by using as features the seismic tremor recorded at a pair of seismic stations, selected with the simple criteria that they should not be located in the very near of the summit craters, since they can be

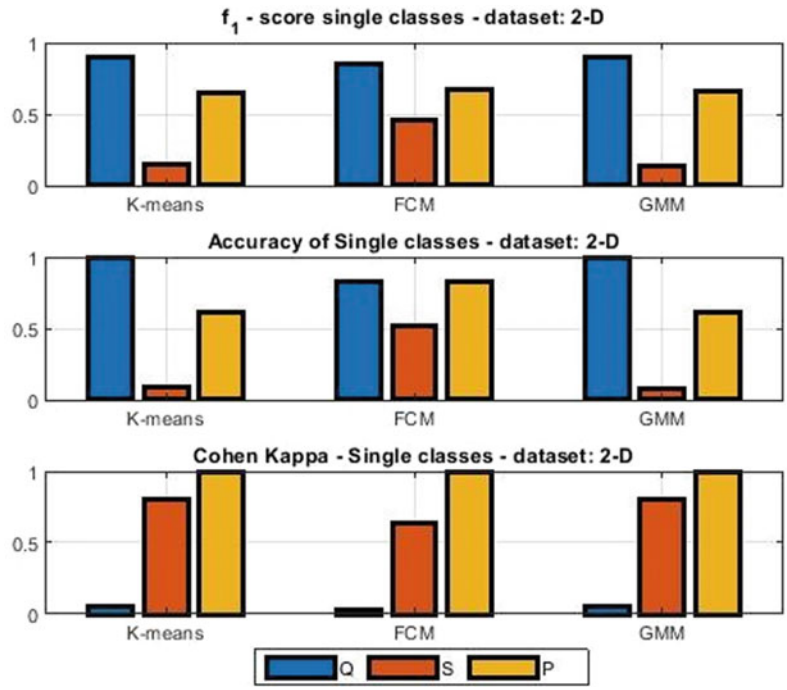


Fig. 15 Comparison among the unsupervised classifiers in terms of f_1 for each individual class by using the 2-D dataset (Nunnari 2021)

affected by spot phenomena. In more detail, the classification regions obtained by clustering the tremor recorded at the ESLN and EBCN stations, by the three considered unsupervised classifiers, are shown in Fig. 14.

It can be seen that the three considered algorithms determine well-distinct regions: the one in the lower corner, in red colour, characterized by low values of the tremor at both stations, resemble the **Q** class, the intermediate one in green the **S** class and, finally, the region in blue colour, the **P** class. As mentioned above, assuming as true the time series of classes provided by the expert volcanologist, it was possible to compute the f_1 score, the Accuracy and the Cohen Kappa indices for each individual class, as shown in Fig. 15.

It is possible to see that the f_1 score and the *Accuracy* for class **Q** and P are quite high regardless of the clustering algorithm, but the same is not true for the class **S**. These achievements are also supported by the Cohen K index. In conclusion, results show that samples of the P and to less extent the Q class samples are clustered with reliable accuracy, while samples of class S are frequently confused as Q and or P, thus indicating that the transition between Q and S or S and P is not clearly discriminated on the basis of the seismic tremor level only.

10 Clustering Multivariate Geophysical Data by Using SOM

The aim of this section is to describe the application of Self Organized Maps (see Sect. 7.2 in Chapter "Machine Learning: The Concepts") to cluster a multivariate geophysical dataset recorded by the monitoring networks at Mt Etna during 1996–2003. The motivation was that some authors have hypothesized that the eruption occurred at Mt Etna in 2001 was a change point in the regime of volcanic activity (Burton et al. 2005). Indeed, after 2001 most eruptive activity has concerned only the flanks of the volcano rather than summit craters area. To be more precise, after 1997 the Mt Etna area inflated with a deformation rate that progressively reduced with time, nearly vanishing between 1998 and 2000. Moreover, low-eruptive rate summit

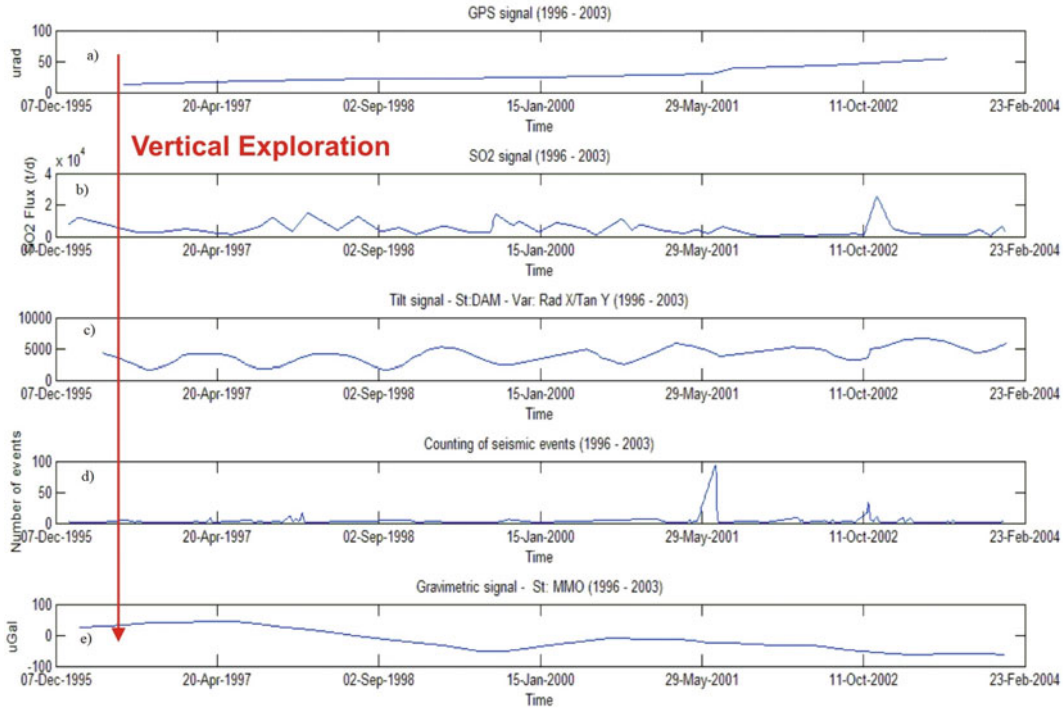


Fig. 16 The multivariate time series: **a** area dilatation, **b** flux of SO_2 emissions, **c** tilt signal recorded at the DAM station, **d** number of seismic events and **e** gravimetric data recorded at the station MMO (Di Salvo et al. 2013)

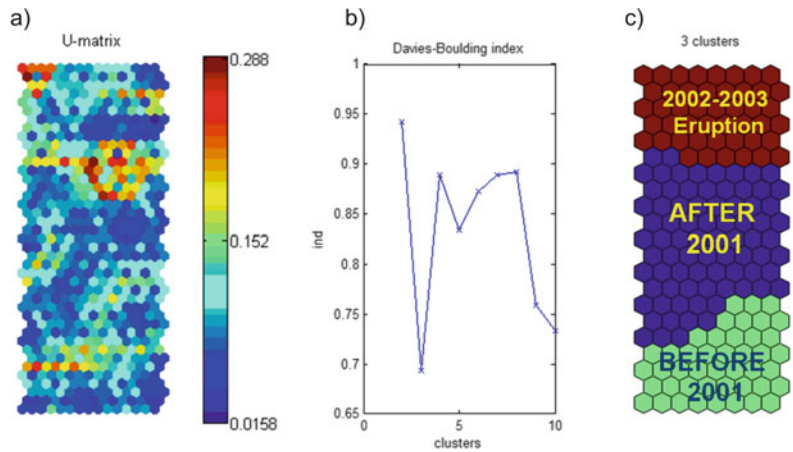


Fig. 17 **a** U-matrix after the training process. **b** Davies–Bouldin performance index. **c** Final cluster lattice structure, which, according to the DB index, suggests that observations best clusters into three classes (Di Salvo et al. 2013)

eruptions occurred, punctuated by lava fountains. After 2001, Etna deflated, feeding higher-eruptive rate flank eruptions, along with large displacements of the entire East-flank. In order to support this conjecture (Di Salvo et al. 2013) proposed to cluster the data set shown in Fig. 16, which consists of ground deformation (area dilatation), obtained from GPS data, the flux of SO_2 emissions, the tilt signal recorded at the DAM station, the number of seismic events and the time series of gravimetric data recorded at the station referred to as MMO.

Such a dataset was used for training a SOM, whose features are shown in Fig. 17a in terms of

the so-called Unified Distance Matrix, shortly referred to as the U-matrix, which provides a colour-matrix that represents distances between neighbouring map units. The obtained U-matrix points out the cluster structure of the map, with high values indicating the cluster border and uniform low values areas indicating the clusters.

In order to evaluate the optimal number of clusters recovered by the trained SOM, the Davies–Bouldin criterion (Davies and Bouldin 1979), was adopted. Such a criterion consists in evaluating an index which is defined as a function of the number of clusters and the within-cluster distance, as described by the following expression:

$$DB = \frac{1}{n}\sum_{i=1}^{n} \max_{i \neq j} \left\{ \frac{S_n(Q_i) + S_n(Q_j)}{S_n(Q_i, Q_j)} \right\} \qquad (8)$$

where n is the number of clusters, S_n is the average distance of all cluster objects to their cluster centers and $S_n(Q_i, Q_j)$ the average distance between clusters Q_i and Q_j. Small values of the DB indices indicate that the clusters are compact and their cluster centers are far away. For the considered dataset the DB index computed versus the number of clusters is shown in Fig. 17b, which supports the conclusion that observations of the considered dataset are best clustered into three classes, as shown in Fig. 17c. In this last figure labels were associated to the corresponding time series, which allowed us to understand that the three recognized clusters corresponding to observation performed before 2001, after 2001 and during 2002–2003. During this last time interval an eruption occurred on Mt Etna.

References

Allen RV (1978) Automatic earthquake recognition and timing from single traces. Bull Seismol Soc Am 68 (5):1521–1532

Allen RM, Kanamori H (2003) The potential for earthquake early warning in southern California. Science 300:786–789

Bentez MC, Ramrez J, Segura JC, Ibanez JM, Almendros J, Garca-Yeguas A, Cortes G (2006) Continuous HMM-based seismic-event classification at Deception Island, Antarctica. IEEE Trans Geosci Remote Sens 45(1):138–146

Bertucco L, Coltelli M, Cristaldi A, Mangiagli S, Nunnari G, Pecora E (1999) Automatic classification of eruptive events by the VAMOS system. In: Proceedings of the IV GIAST workshop, Sansepolcro (Italy), Sept 1999

Burton M, Neri D, Andronico S, Branca T, Caltabiano S, Calvari RA, Corsaro P, Del Carlo G, Lanzafame L, Lodato L, Miraglia F, Mur G, Salerno, Spampinato L (2005) Etna 2004–05: an archetype for geodynamically-controlled effusive eruptions. Geophys Res Lett 32:L09303. https://doi.org/10.1029/2005GL022527

Cannavò F, Cannata A, Cassisi C, Di Grazia G, Montalto P, Prestifilippo M, Privitera E, Coltelli M, Gambino S (2017) A multivariate probabilistic graphical model for real-time volcano monitoring, on Mount Etna. J Geophys Res Solid Earth 122:3480–3496. https://doi.org/10.1002/2016JB013512

Cassisi C (2016) Probabilistic reasoning over seismic time series: volcano monitoring by hidden Markov models at Mt. Etna. Pure Appl Geophys 173:2365–2386

Corradino C, Ganci G, Cappello A, Bilotta G, Calvar S, Del Negro C (2020) Recognizing eruptions of Mount Etna through machine learning using multiperspective infrared images. Remote Sens 12:970. https://doi.org/10.3390/rs12060970

Curilem M, De Mello RF, Huenupan F, San Martin C, Franco L, Hernndez E, Rios RA (2018) Discriminating seismic events of the Llaima volcano (Chile) based on spectrogram cross-correlations. J Volcanol Geotherm Res 367:63–78

Davies DL, Bouldin DW (1979) IEEE Trans Pattern Anal Mach Intell PAMI-1

Di Salvo R, Montalto P, Nunnari G, Neri M, Puglisi G (2013) Multivariate time series clustering on geophysical data recorded at Mt. Etna during 1996–2003. J Volcanol Geotherm Res 251:65–74

Falcin A (2021) A machine-learning approach for automatic classification of volcanic seismicity at La Soufrière Volcano, Guadeloupe. J Volcanol Geotherm Res 411

Falsaperla S, Graziani G, Nunnari G, Spampinato S (1996) Automatic classification of volcanic earthquakes by using multi-layered neural networks. Nat Hazards 13:205–228

Hajian A, Cannavo F, Greco F, Nunnari G (2019) Classification of Mount Etna (Italy) volcanic activity by machine learning approaches. Ann Geophys 62 (2):231

Hibert C, Mangeney A, Grandjean G, Baillard C, Rivet D, Shapiro NM, Crawford W (2014) Automated identification, location, and volume estimation of rockfalls at Piton de la Fournaise volcano. J Geophys Res Earth Surf 119(5):1082–1105

Khosravikia F, Clayton P (2021) Machine learning in ground motion prediction. Comput Geosci 148

Kong Q, Trugman DT, Ross ZE, Bianco MJ, Meade BJ, Gerstoft P (2019) Machine learning in seismology:-turning data into insights. Seismol Res Lett 90

Langer H, Falsaperla S, Messina A, Spampinato S, Behncke B (2011) Detecting imminent eruptive activity at Mt Etna, Italy, in 2007–2008 through pattern classification of volcanic tremor data. J Volcanol Geotherm Res 200:1–17

Langer H, Falsaperla S, Hammer C (2020) Advantages and pitfalls of pattern recognition—selected cases in geophysics. Elsevier, Amsterdam, pp 1–331

Langet N, Maggi A, Michelini A, Brenguier F (2014) Continuous kurtosis-based migration for seismic event detection and location, with application to Piton de la Fournaise Volcano, La Réunion. Bull Seismol Soc Am 104(1):229–246

Lin TL, Wu YM (2010) Magnitude estimation using the covered areas of strong ground motion in earthquake early warning. Geophys Res Lett 37:L09301

Lin J, Keogh E, Wei L, Lonardi S (2007) Experiencing SAX: a novel symbolic representation of time series. Data Min Knowl Disc 15:107–144

Malfante M, Dalla Mura M, Mars JI, Mtaxian JP, Macedo O, Inza A (2018a) Automatic classification of volcano seismic signatures. J Geophys Res Solid Earth 123(12):10645

Malfante M, Dalla Mura M, Mtaxian JP, Mars JI, Macedo O, Inza A (2018b) Machine learning for volcano-seismic signals: challenges and perspectives IEEE Signal Process Mag 35(2):20–30

Nakamura Y (1988) On the urgent earthquake detection and alarm system (UrEDAS). In: Proceedings of 9th world conference of earthquake engineering, vol VII, pp 673–678

Nunnari G (2021) Clustering activity at Mt Etna based on volcanic tremor: a case study. Earth Sci Inform 14:1121–1143. https://doi.org/10.1007/s12145-021-00606-5

Reddy R, Nair RR (2013) The efficacy of support vector machines (SVM) in robust determination of earthquake early warning magnitudes in central Japan. J Earth Syst Sci 122(5):1423–1434

Ross AE, Meier MA, Hauksson E, Heaton TH (2018) Generalized seismic phase detection with deep learning. Bull Seismol Soc Am 108:2894–2901

Simons FJ, Dando DE, Allen RM (2006) Automatic detection and rapid determination of earthquake magnitude by wavelet multiscale analysis of the primary arrival. Earth Planet Sci Lett 250(12):214–223

Titos M, Bueno A, Garcia L, Benitez C (2018) A deep neural networks approach to automatic recognition systems for volcano-seismic events. IEEE J Sel Top Appl Earth Obs Remote Sens 11(5):1533–1544

Tsang L, Allen RM, Wurman G (2007) Magnitude scaling relations from P-waves in southern California. Geophys Res Lett 34:L19304

Wurman G, Allen RM, Lombard P (2007) Toward earthquake early warning in northern California. J Geophys Res 112:B08311

Deep Learning: The Concepts

Abstract

This chapter presents the concepts of Deep Learning (DL) models. Also, comprehensive descriptions of general DL architectures are provided, and different types of such models are categorized and discussed in an applicable and reproducible form. The main feature of this chapter is presenting various examples with *python* codes so that the reader can easily follow the algorithms of designing and testing DL models step-by-step.

1 Introduction

As a subfield within Machine Learning (ML) methods, Deep Learning models, which are also known as deep neural networks, imitates the procedure of learning in humans and as depicted in Fig. 1, is generally implemented based on three steps: providing dataset(s), learning from them and finally getting the desired result(s).

In the learning step, the data are represented based on the features, which may be pre-defined (e.g., through a statistical function) or calculated and determined automatically by the model itself. In the latter case, the method decides whether a feature "is" or "is not" appropriate to output the target based on the data elements. In many DL models, e.g., in convolutional neural networks, feature extraction is handled automatically in a procedure known as "representation learning".

Later in this chapter, comprehensive descriptions of general DL architectures are provided, and different types of such models are categorized and discussed in applicable and reproducible forms.

2 Deep Learning, an Overview

Earlier in Chapter "Machine Learning: The Concepts" general architecture of an ANN model was discussed. It must be clarified there are no considerable differences between a DL and a shallow ANN model in terms of elements and connections and the basic usages of the neurons. The main variations arise from the structure and volume of the two networks. Other differences arise from multi-input, multi-layer, and multi-output structures that result in more flexible and adaptable models (Alzubaidi et al. 2021). The general architecture of a simple deep MLP (usually called DNN) is illustrated In Fig. 2. The size and amount of the hidden layers have to be considered as the main difference from a shallow model. Tens of layers, each of which, consisting of hundreds of neurons may be used in a simple DL model.

The main elements of such DL network are as follows (Aggarwal 2018):

- Input layer(s): Could be in any form of divers datasets like timeseries, multidimensional

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

A. Hajian et al., *Intelligent Methods with Applications in Volcanology and Seismology*, Advances in Volcanology, https://doi.org/10.1007/978-3-031-15432-4_4

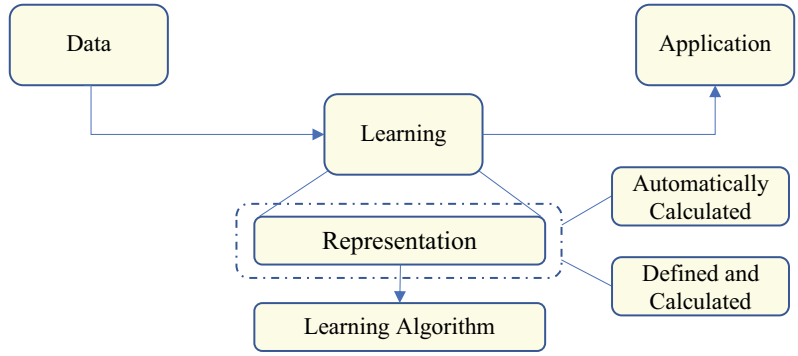


Fig. 1 The three successive steps of a typical ML method

feature specified data, time-dependent streams and even any combination of the mentioned datasets. New Inputs may include outputs from previous models or layers (in functional architectures).

– Hidden layers: As the representation core of the network, in this block, using a set of a relatively massive number of layers (compared to the shallow ANNs) and neurons, the feature extraction is handled. The input data will pass through the set of layers (filters) and in each layer, a group of features is determined. Mentioned features will be passed to the next layer successively, and, at the output layer, the target will be simulated and based on the learning algorithm and the considered cost function, the learning procedure will be continued to achieve the optimum output.

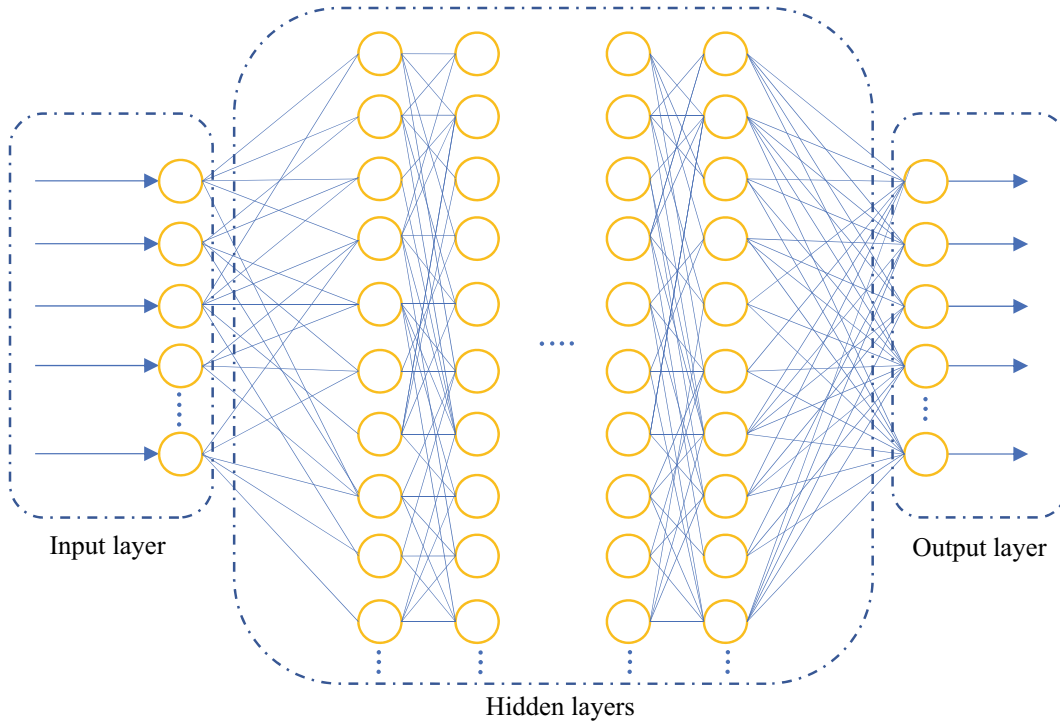


Fig. 2 Schematic architecture of a deep multi-layered neural network

– Output layer: This layer coalesces the current values of the last neurons and produces the final result(s) using current weights in each connection. As another difference with most ANN algorithms, the size of the output is not restricted to the "one" and this may be facilitative and useful in solving many geophysical data problems.

Although the architecture demonstrated in Fig. 2, is basically the same as an ANN, the details in DL networks are sources of many differences in applications, performances and even the procedure of feeding data to the network. In fact, the deep difference here with respect to a shallow model, arises from the "deep" representing process of the network which explores the input data as a model, based on the features, each of which where extracted by every single layer in the hidden layers sequences. The concept is more understandably applied in (deep or shallow) convolutional neural networks as a revolutionary idea.

Images are usually thought of as a two-dimensional matrix of numbers. Each element in this two-dimensional matrix is equivalent to one pixel. If we have a 100 × 100 image, this means we have 10,000 pixels. They are next to each other regularly. Now imagine we want a layer for this 10,000 elements structure. Consider 10,000 neurons for an MLP network input. Adding more neurons and layers to this MLP network makes our model include a large volume of parameters; its calculations are costly, and the possibility of overfitting increases.

As demonstrated in Fig. 3, when convolutional operators are performed on a Multi-angle Imaging SpectroRadiometer (MISR) data as input and translated (represented) versions of it are achieved, some of which, are more related to the desired output (e.g., classification or edge detection problems) and the related weights are regulated so that the loss function become minimized.

3 Deep Learning, Pros and Cons

The first advantage of using DL networks is that, there is no need to introduce and calculate features and they are implicitly determined automatically by the network. The second advantage is that the learning procedure could be achieved through a multi-level hierarchical framework in DL models. This concept provides informative and easily accessible keys to the network options. The corresponding features show how the learning is progressing from the low to high levels in the network.

The accuracy of problem-solving is the next benefit of using DL networks. A properly defined and learned DL model usually is more precise than other ML solutions.

The main reason an ML algorithm is used, is the generalization approach. The model is trained mainly to have answers for other samples not involved in the training data set. From this point of view, DL networks are ranked definitely in the top places of ML solutions.

Aside from mentioned inherited advantages, because of being placed among newly developed methods and also providing acceptable results in many problems, DL software packages and hardware are being designed so fast. Using DL networks is possible in today's desktop PCs, and many libraries are freely available on the internet in this context. Many professional groups on various social platforms provide helpful solutions and guidance on any related topics. The world's largest companies and brands are performing investments in the field, and the DL concept is considered to be among the hottest subjects of science and technology.

Regarding "cons", "challenges", or "limitations", there are some items one could suffer from when using DL methods. Theoretically and analytically, good and descriptive mathematical relations are not provided in the field and current achievements and developments are based on trial and error studies in some cases.

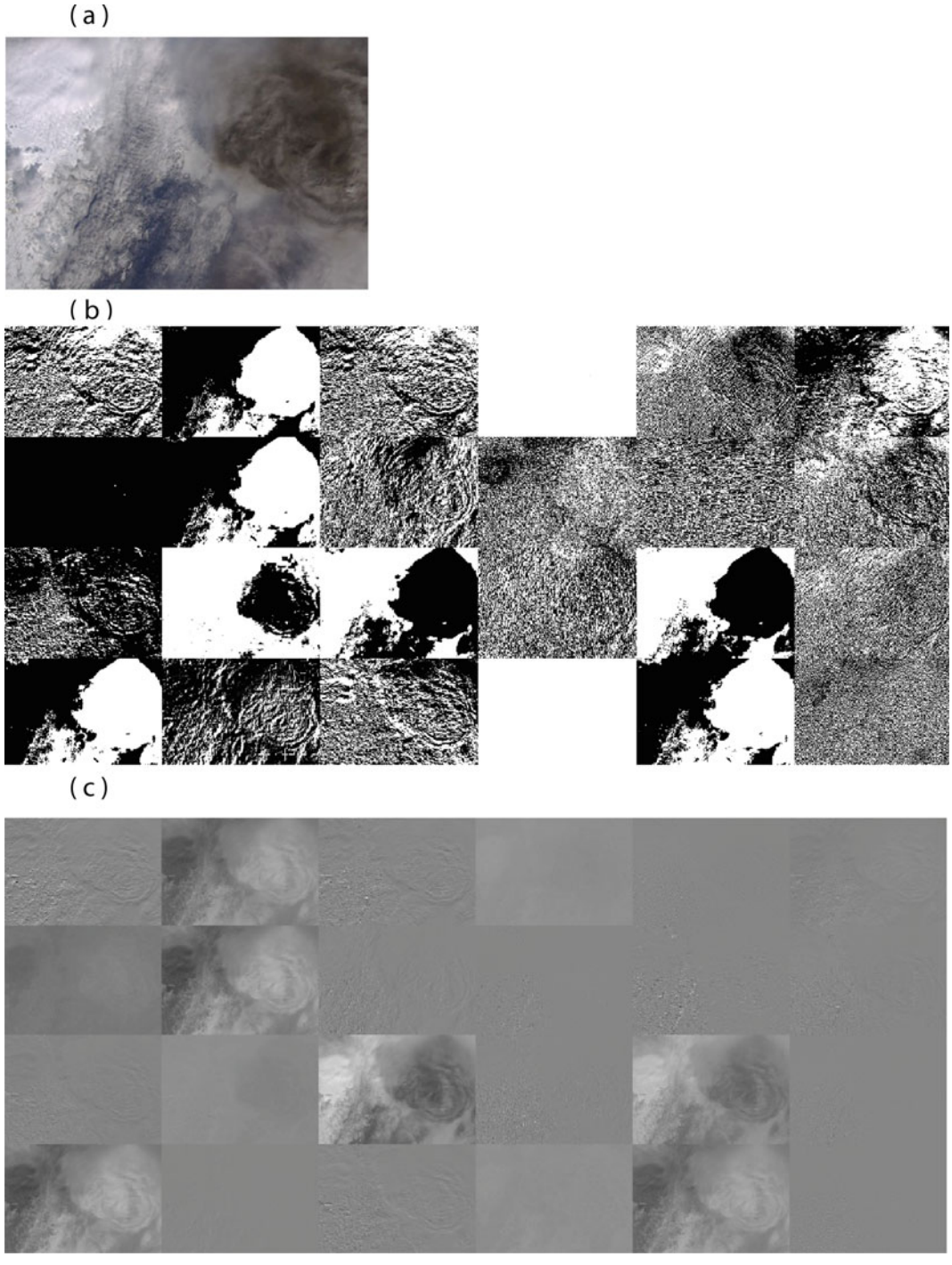


Fig. 3 **a** A MISR image captured reflection section (NASA image courtesy Jeff Schmaltz, MODIS Rapid Response Team at NASA GSFC) and **b** represented features extracted using random kernels. **c** Normalized grayscale images of the previous outputs

The next disadvantage originates from the volume of the DL networks and that is, they consume more time and computational resources compared to other ML methods. DL models need large datasets for accurate model discrimination. In some disciplines, such as medical imagery or oil exploration, where, data acquisition is expensive or time-consuming, the lack of such databases could result in poor performance.

Many parameters may affect the output of a DL model. Determining the optimum values is not usually done through rule-based solutions and the problem is generally handled using empirical methods.

The last negative point that should be categorized as ML models' general problem is related to the learning step. The network could easily behave like an overtrained or undertrained network and again, no analytical solution is provided for this problem.

4 Layers in a Deep Learning Models

An array of neurons, sets of them as layers, operators and combinations of all mentioned in different blocks, are used in a DL model (Zhang et al. 2021). In most cases, they are connected sequentially in the architecture, and in some models, called functional structures, side or parallel connections are utilized. Blocks of layers have specific roles and significance such as input data calling, feature extraction, data dimensions changing and output composer.

The possibilities with layers in a DL model are relatively enormous, which is undoubtedly a source of enrichments in the approach. Some significant standard layers are described in Table 1. Other types of layers may also be used in specific models, some of which will be discussed in the sections related to the corresponding DL algorithms in the following sections.

Table 1 DL major layers

Name	Description, functionality and significance
Input	Based on the characteristic of the problem, the input layer may be fed by any form or combination of multidimensional data, even time-dependent elements
Batch and layer normalization (standardization)	Normalizes the batches or entire layer of elements across desired dimensions. This rescaling is used for the stability of the solution and to speed up the training procedure
Convolutional	As the core of a convolutional neural network, the layer represents the data from the previous layer in terms of new features using convolution operators (each filter detects a particular feature). This layer could be sets of one-, two- or three-dimensional blocks or combinations of them
Pooling	Down-samples the input data by dividing the input space into pooling sub-regions and computes the average or maximum of each sub-region and it is often used after convolutional layers
Embedding	Is used for vectorizing categorical features (positive integers such as indexes)
Fully connected (dense)	The layer provides a vectorized representation of the previous layer and is used mainly for converting the target of the previous layer to a specific set of nodes such as in classification problems. All nodes in the layer, are connected to all inputs of the prior layer and that is why it is called a fully connected layer
Flatten	Flattens the input with the given dimensions into the channel form
Activation	The layer applies an activation function to the input data to handle the relation between input and target data complexity
Dropout	By randomly setting some input elements to zero, this layer aims to prevent overfitting that could directly affect the performance of the network generalization
Output	This layer produces the network result (output) based on optimization and loss calculation algorithms

5 Deep Learning Models

As stated before, learning algorithms are different in various ML methods and this diversity is also present in regarding DL architectures. From a general point of view, there are four different methods for performing the learning procedure in a DL model which are: supervised, unsupervised, semi-supervised and reinforcement learning. Among mentioned, the first two are considered as the main widely used methods and will be discussed here in details.

5.1 Supervised Deep Learning Methods

Guided with the help of human knowledge (or experience), in a supervised DL method, as well as, input parameters, the problem's solution for each example is fed to the network as the target. Hence, The provided set of inputs-targets will base the knowledge for the feature extraction step. The applications of the supervised method include data classification, segmentation, processing, regression, prediction and labeling.

5.2 Deep Convolutional Neural Network

The most famous and widely used supervised DL model is definitely, Convolutional Neural Network (CNN). CNN models have been proven to provide many state-of-the-art solutions and achievements for different applications. Using convolutional and pooling layers in the architecture is the major distinctive specification of a CNN model. For better comprehension, the 5th generation of the first successful CNN network (LeCun et al. 1989), is described in detail here. This CNN is referred to as LeNet-5 or, simply, LeNet (Fig. 4). The network was designed initially for digitalizing handwritten numbers and was successfully applied, distinguishing handwritten zip code numbers (LeCun et al. 1998). The details about each of the seven layers of the LeNet are as below:

- Inputs: The inputs consist of 32 by 32 pixel, grayscale images normalized between 0 and 255.
- L1: This convolutional layer consists of six blocks with convolutional kernels of size 5 * 5. As a result, the feature map (the result of performing convolution filter to the input

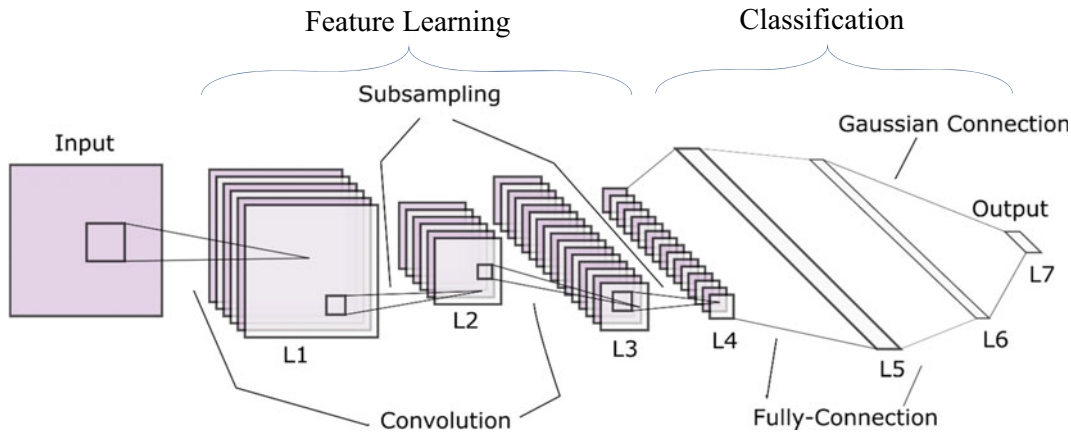


Fig. 4 LeNet proposed by Lecun in 1990. This network is the 5th generation of the first successful CNN model proposed

instances) will be 28 * 28. Considering padding (0) and stride (1) sizes, 28, is the result of: [(input size − kernel size + 2 × padding)/stride] + 1. The mentioned dimensions guarantee that the input pixels will not fall out of the boundaries of the convolution kernel.
- L2: In this subsampling (pooling) layer, the 28 * 28 feature maps are subsampled (downsampled) to 14 * 14 maps using 2 * 2 windows. This transformation is performed to remove the network's position reliance. In other words, because after mapping the features from an image by convolution, the resulting information corresponds to the positions of the components within the image, downsampling can weaken this effect and make the features bolded within the map by their structures and not their positions.
- L3: Another convolutional layer that contains 16 blocks that resulted from performing 5 * 5 convolutional kernels to 14 * 14 maps in layer L2. Hence, the produced feature maps will be of sizes 10 * 10.
- L4: Similar to layer L2, this layer subsamples the L3 layer with 2 * 2 windows resulting in 5 * 5 outputs.
- L5: This is another convolution layer with 120 kernels of size 5 * 5. Since the output of L4 is also the same size, the result will be of size 1 * 1. L4 is fully-connected to L5 in this structure as a dense connection (this is an option and is not a general rule).
- L6: Again, a fully connected layer joined to all L5 nodes.
- L7: Using the euclidean radial basis functions for each class, the L6 outputs is classified in this layer by calculating the errors between the targets and the model outputs. Finally, the ten numbers corresponding to the digits from 0 to 9 are achieved in this layer.

Although convolutional networks differ from the relatively simple LeNet model, the basics are the same. Based on the reviewing LeNet, it may be concluded that in a CNN model:

- Multiple Sequences of convolutional and pooling layers are used.
- In each convolutional layer, convolutional filters are applied with different sizes (not mandatory).
- There may be blocks with different filters in each layer.
- CNN models are suitable for processing and extracting spatial features within datasets such as images, 2D and 3D data and any composition of them.

Let us now discover the foundation of a CNN model by an example in the next section.

5.3 Image Classification by CNN: Fault Detection in Synthetic Seismic Data

A set of 1000 seismic gathers of size 128 by 128 pixels, are extracted from a main 1024 by 1024 synthetic gather which includes four faulting mechanisms (Fig. 5). The goal is to find the accurate number of faults in each sub-image (Fig. 5b). This section describes the experiment in detail to provide a comprehensive perception of using a typical convolutional model.

Examples provided in this chapter are coded using the Keras (Gulli and Pal 2017) environment which is an open-source library developed to enable deep neural networks experimentations. Keras's development, an Application Programming Interface (API) and a TensorFlow component, is based on simplicity, flexibility, and power. The Keras, enables to define models in both sequential and functional modes which will be described later. Minimum requirements for using the library are stated officially to be:

- Python 3.6–3.9
- Ubuntu 16.04 or later
- Windows 7 or later
- macOS 10.12.6 (Sierra) or later.

For the ease of reproduction and testing by the respected readers, the experiments and the corresponded codes are written in Google Collaboratory or simply, colab notebooks (Bisong 2019). It must be noted that the Tensorflow and Keras

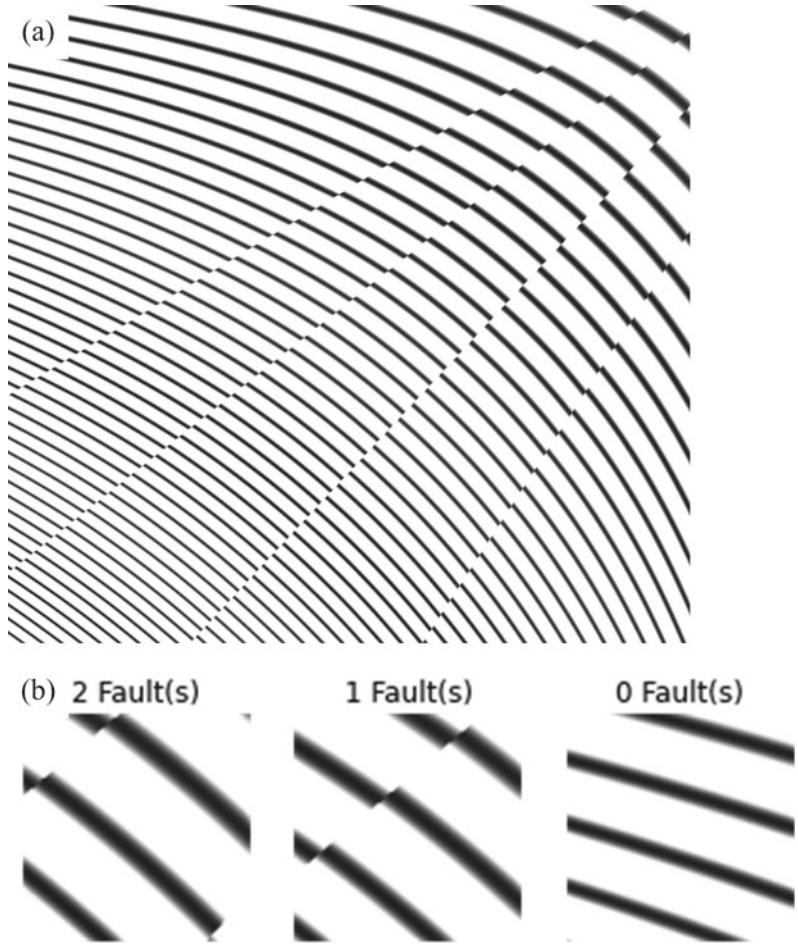


Fig. 5 **a** A synthetic 1024 by 1024 pixels seismic synthetic gather including four faulting mechanisms. From this gather, a set of 1000 randomly chosen 128 by 128 gathers are extracted and used for determining number of faulting mechanisms. **b** Three samples randomly selected from each class directory

APIs are both fully pre-installed on the colab environment and usually, no pre-configuration is needed for general usages.

In the following, the main parts of the codes are formatted normally, while the outputs that are followed after ">" sign, are bolded for better discrimination. Also, the in-cell comments are provided next after the "#" signs (as Python's default commenting method). Some of the unnecessary parts of the outputs, e.g. the training procedure of the models per some epoches, are cropted to avoid text clutter.

First, as the code runs in google colab, the google drive is mounted in the notebook to load the train and test datasets:

```
from google.colab import drive
drive.mount(''/content/drive'')
>Mounted at /content/drive
```

It is customary in the first step to call all the necessary libraries and modules which are going to be used in the following steps. Some of these are installed by default in the Colab environment and some, have to be installed by the user. The user can check the list of installed modules by pip command ("pip list"). Related description for each module is provided in the first use:

```
Import random
import os
import numpy as np
```

```
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import preprocessing
from tensorflow.keras.preprocessing import
  image_dataset_from_directory
from keras.preprocessing.image import
  ImageDataGenerator, load_img
from keras.models import Sequential
from keras.layers import Conv2D, Dense,
  Flatten, MaxPooling2D, Dropout
from keras.preprocessing import image
```

In the Keras API, one can use a separated folder structure for training and validating datasets or putting them together and letting Keras to split them randomly. In this example, all seismic gathers are separated based on the number of fauls detected in the gather and here, using "os", "random" and "matplotlib" modules, an instance from each folder is loaded and plotted Fig. 5b. Hence, while loading the data, labels are also loaded in the categorical format:

```
tr_data_path = ('/content/drive/MyDrive/
   syndata/syn128_1000/')
#loading the directory folders for
   extracting the lables
lbls = os.listdir(tr_data_path)
plt.figure()
cnt=0
#walking through each sub-directory for
   counting the inner files and
#selecting one of them.
for k in lbls:
   os.chdir(tr_data_path+k+'/')
   rnd_idx = random.randint(0, len(os.
   listdir()))
   img = load_img(os.listdir()[rnd_idx])
   ax = plt.subplot(1, len(lbls), cnt + 1)
   plt.imshow(img)
   plt.title(lbls[cnt]+' Fault(s)')
   plt.axis(''off'')
   cnt +=1
```

Any DL network must be defined in Keras API at the very first step. This procedure is layer and order based. After crating a "Sequential" model, the layers should be introduced step by step with mandatory and optional parameters. in the first layer, the input images could be resized using "input_shape" parameter. Each convolutional layer is defined with the chosen "activation" function. for example, the third convolutional layar consists 64 different 3 by 3 convolutional filters with "relu" activation functions. For a descriptive and comprehensive guide on using different activation functions, the respected readers are referred to the research presented by Agostinelli et al. (2014). The utilized architecture in this experiment is illustrated in Fig. 6:

```
CNN_Model = Sequential()
CNN_Model.add(Conv2D(16,(7, 7), input_
   shape=(128, 128, 1),activation='relu'))
CNN_Model.add(MaxPooling2D(pool_size=
   (2, 2)))
CNN_Model.add(Conv2D(32,  (6,6),  activa-
   tion='relu'))
CNN_Model.add(MaxPooling2D(pool_size=
   (2, 2)))
CNN_Model.add(Conv2D(64,  (3,3),  activa-
   tion='relu'))
CNN_Model.add(MaxPooling2D(pool_size=
   ((2, 2)))
```

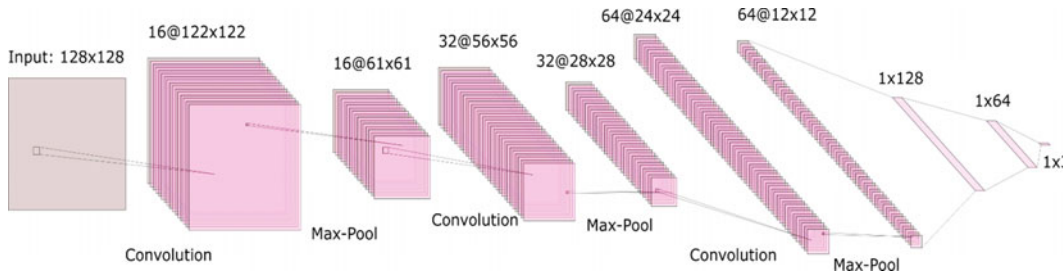


Fig. 6 The architecture of the CNN model used for image classification

```
CNN_Model.add(Flatten())
#Construction of the fully connected layers
CNN_Model.add(Dense(units =128, activa-
  tion='relu'))
CNN_Model.add(Dense(units =64, activa-
  tion='relu'))
CNN_Model.add(Dense(units =3, activa-
  tion='softmax'))
CNN_Model.compile(optimizer='rmsprop',
loss='categorical_crossentropy', met-
rics=['accuracy'])
#Splitting the data into the "train' and
  "validation" subsets. The "labels'
#parameter is set to "inferred", which
  means the labels are automatically saved
#from the names of the folders. Also, the
  "batch_size" indicates number of
#samples that are fed to the network
  simultaneously.
trn_set = image_dataset_from_directory
  (tr_data_path, labels = 'inferred',
  validation_split = 0.20, subset =
  "training", seed=123, label_mode
  ='categorical', batch_size = 32, color_
  mode = "grayscale", image_size =
  (128, 128))
val_set = image_dataset_from_directory
  (tr_data_path, labels = 'inferred',
  validation_split = 0.20, subset =
  "validation", seed=123, label_mode
  ='categorical', batch_size = 32, color_
  mode = "grayscale", image_size =
  (128, 128))
#Training the Model using 20 epochs
history = CNN_Model.fit_generator(trn_set,
  steps_per_epoch=800//32,
epochs=20, validation_data=val_set,
  validation_steps=200//32)
```

```
>Found 1000 files belonging to 3 classes.
Using 800 files for training.
>Found 1000 files belonging to 3 classes.
Using 200 files for validation.
>Epoch 1/20
25/25 [==============================] - 2s 56ms/step
- loss: 57.1058 - accuracy: 0.5013 - val_-
loss: 0.7709 - val_accuracy: 0.7240
Epoch 2/20
25/25 [==============================] - 2s 51ms/step
- loss: 0.5608 - accuracy: 0.7867 - val_loss:
1.5390 - val_accuracy: 0.7500
Epoch 3/20
25/25 [==============================] - 2s 50ms/step
- loss: 2.2589 - accuracy: 0.6641 - val_loss:
0.3512 - val_accuracy: 0.8906
Epoch 4/20
25/25 [==============================] - 2s 51ms/step
- loss: 1.3228 - accuracy: 0.8691 - val_loss:
0.6743 - val_accuracy: 0.8438
Epoch 5/20
.
.
.
25/25 [==============================] - 2s 53ms/step
- loss: 0.4223 - accuracy: 0.9447 - val_loss:
0.9991 - val_accuracy: 0.9219
Epoch 20/20
25/25 [==============================] - 2s 51ms/step
- loss: 0.0205 - accuracy: 0.9907 - val_loss:
1.3692 - val_accuracy: 0.9375
```

The network learning "Loss" and "Accuracy" quantitative parameters can be used to evaluate the performance of the determined model and decide whether to continue or abort the learning process or change the input parameters. The calculated loss and accuracy of the trained model are shown in Fig. 7:

```
print(history.history.keys())
>dict_keys(['loss', 'accuracy', 'val_-
loss', 'val_accuracy'])

#plotting Loss values per epoch
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['loss', 'val_loss'],
  loc='upper left')
plt.show()
#plotting Accuracy values in each epoch
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accu-
  racy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
```

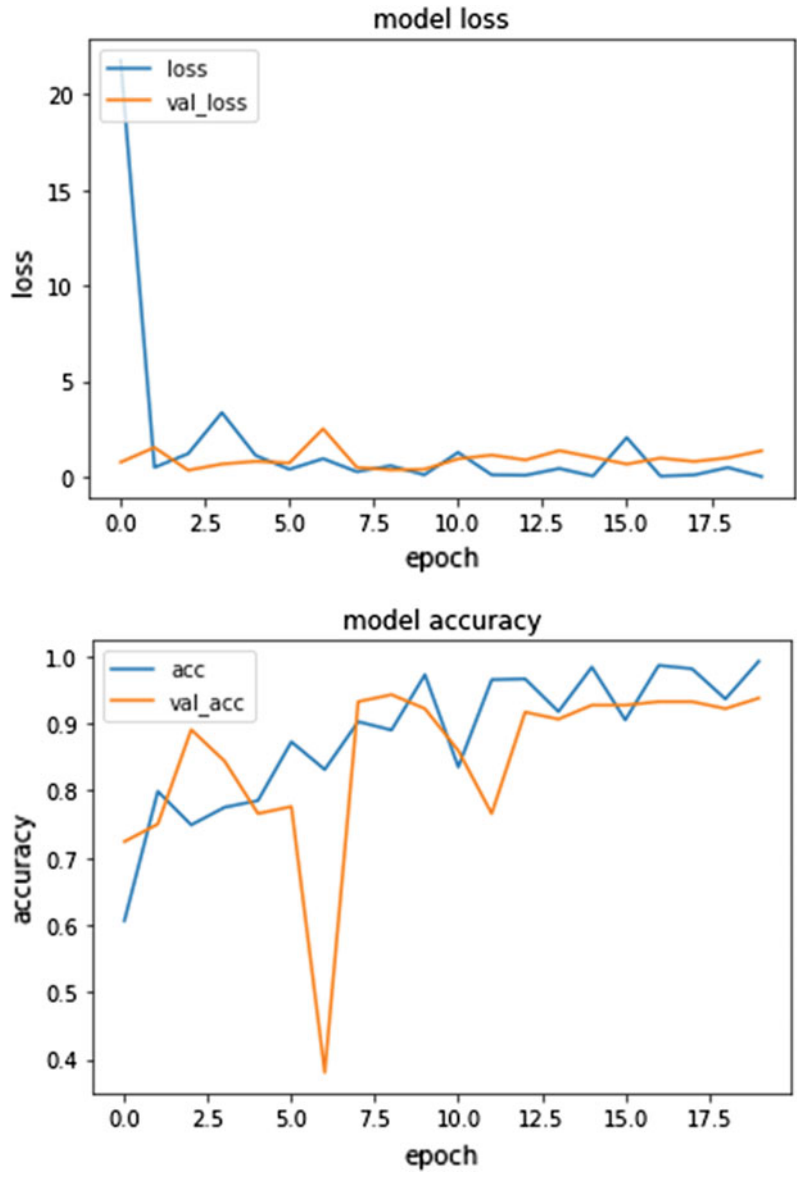


Fig. 7 Calculated loss and accuracy per epoch for the training (top) and validation (bottom) datasets

```
plt.xlabel('epoch')
plt.legend(['acc','val_acc'],
   loc='upperleft')
plt.show()
```

In the last step of the experiment, the trained network is generalized to a set of 24 images (Fig. 8) that are not involved in the "train" and "validation" subsets and were generated separately. Because of the resulting performance of the model, the output of the mentioned prediction is also good, and all predictions are correct except the one in the first column of the third row where there is a fault traced in the upper leftmost of the image while the corresponding output resulted from the model evaluation is "0":

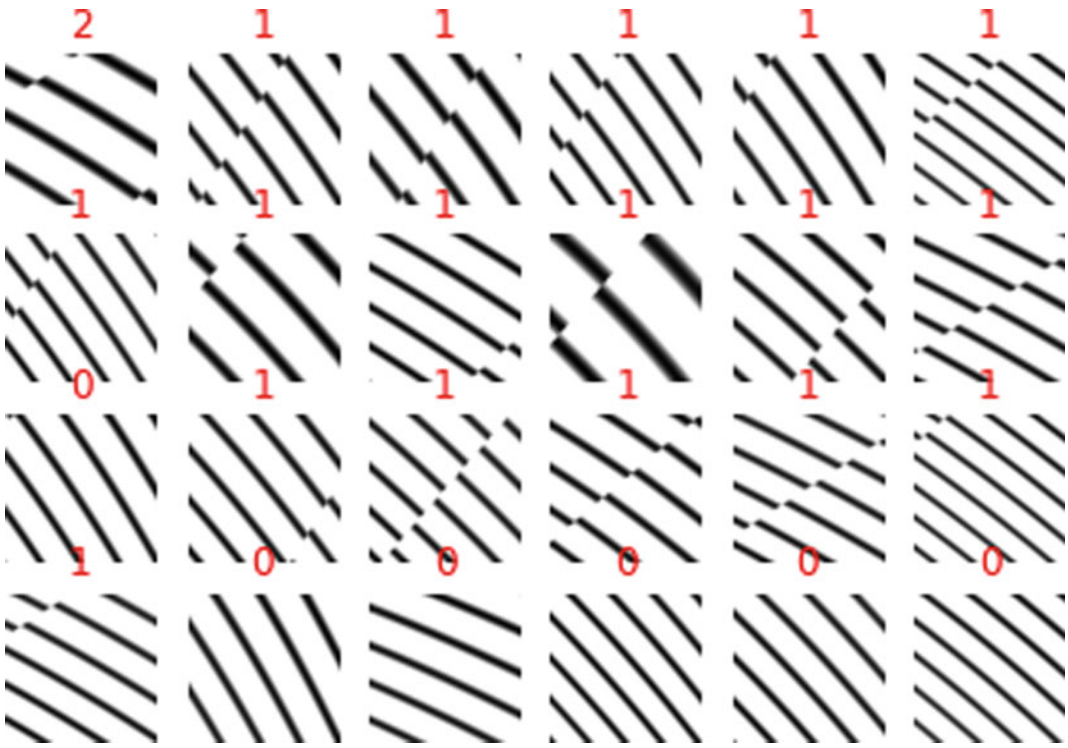


Fig. 8 The results of eveluating the trained model over 24 test sections which are not involved in model training

```
test_data_path = ('/content/drive/
  MyDrive/syndata/syn_test/')
plt.figure()
cnt=0
for k in lbls:
  os.chdir(test_data_path+k+'/')
  imgs = os.listdir()
  for kk in imgs:
    inimg = load_img(kk, target_size=
    (128, 128), color_mode="grayscale")
    tstimg = image.img_to_array(inimg)
    tstimg = np.expand_dims(tstimg,
    axis=0)
    #Generalizing the model to each
    sample of the test data
    predictions = CNN_Model.predict
    (tstimg)
    #Calculation of the output of the
    network weights with respect to
    #the "softmax" activation function
    out = tf.nn.softmax(predictions[0])
    ax = plt.subplot(4,6, cnt + 1)
    plt.imshow(inimg, cmap='gray')
    #Label determination
    plt.title(str(np.argmax(out)),
    color='red')
    plt.axis("off")
    cnt +=1
```

In this experiment, a CNN model was utilized in a simple multi-label classification problem. Usually, data preparation and determination of the parameters used in model training are among the challenging issues for using such models. Here, the training and validation datasets were loaded using the "image_dataset_from_directory" function. Instead, one can manually construct mentioned datasets and use them in the "fit_generator" function. Another point to consider is the number of kernels used in each block of the CNN sequence. Despite what is used here, using an ascending order is more customary in the cases where the inputs are relatively more

complex than are in this experiment. To describe this conceptually, a higher number of kernels means the layer can extract more features. By advancing in the model, the resulting outputs (that are inputs of the next layer) get more complex, there so this increment sounds logical (Sultana et al. 2019). Here, a contrasting pattern was used to show that this is not a principle. Even a combination of incremental and decremental structures can be optimal for solving a particular problem.

5.4 Recurrent Neural Networks

In some real-world problems, there is a relation between successive instances (inputs) of a dataset. Unlike the previous example, where the three classes of the data were independent from each other, in a case like random noise attenuation problem in seismic shot gathers, the coherency (or randomness) of the events is determined after investigating a few successive traces (inputs). In the CNN model used in the previous experiment, the main concept that seems to be missed in the model, is a kind of memory that keeps (removes) the useful information of the previous inputs. As a generalization of the feedforward neural network, Recurrent Neural Networks (RNNs) were originally invented to be used for such sequential prediction problems. Different examples using RNN may include one to one or many, and many to one or many problems. A many to many RNN architecture is depicted in Fig. 9 for example. In the figure, each of input or output layers are in the form of multi-nodes.

The main difference in the architecture of an RNN network is the existence of a looping mechanism which makes it possible for each state to have the previous output(s) fed into the network with current inputs. In other hand, the RNNs are equipped with a memory and can recall (or forget) previous outputs which are reflowed in the network for sequential prediction with new weights as below:

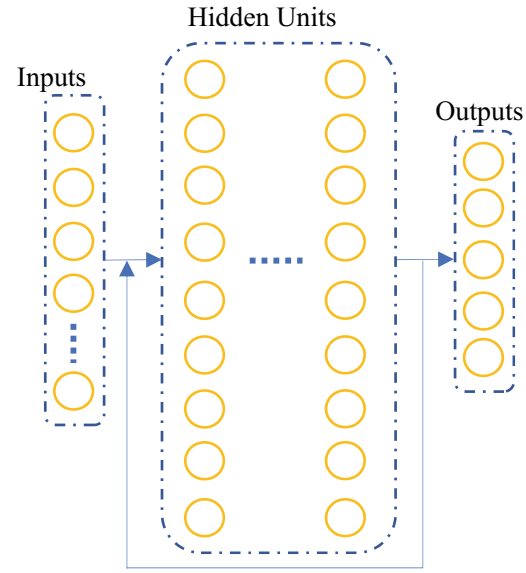


Fig. 9 Schematic illustration of a "many to many" RNN model in a general. form

$$h_t = f_w(h_{t-1}, x_t) \tag{1}$$

where h and x stand for state and inputs respectively and f_w is the weight function.

RNNs also are used in convolutional models mainly to improve feature extraction from sequential data e.g. movie's successive scenes.

As a disadvantage, RNNS suffer from gradient vanishing and exploding, making it difficult to handle learning procedures, especially in long data sequences where parameter updatings become unuseful and meaningless.

General applications of RNNs may include speech recognition, music generation, machine translation and sentiment classification. In earth science, RNNs are widely used in problems such as inversion (Liu et al. 2020), denoising (Colom and Morel 2019), prediction (Bhandarkar et al. 2019), classification (Liu and Sun 2020), and automatization (Cai et al. 2019).

The most successful and commonly used RNN models include Long Short Term Memory, LSTM (Hochreiter and Schmidhuber 1997), Bidirectional Recurrent Neural Networks, BRNN

(Schuster and Paliwal 1997) and also, as a modification to the LSTM networks, the Gated Recurrent Unit, GRU (Dey and Salem 2017) networks. The last one is introduced mainly to reduce the LSTM model's computational complexity. In continuation of this section, after a brief description of LSTM and GRU models, an example of using the LSTM network for a prediction (extrapolation) problem in two-dimensional datasets is presented in detail.

5.5 Long Short Term Memory Network

In the LSTM architecture, there is a mechanism for keeping or forgetting weights in updating them, based on three steps (gates). The forget gate controls what information to throw away from memory or even decides how much of the past information should be remembered. The update, sometimes called the input gate, controls what new information should be added to the cell state from the current input, and finally, the output gate that conditionally decides what to be considered as output. The LSTMs are widely used for time series and time-dependent regression, classification, prediction, and processing problems. This is because there can be lags of unknown duration between essential events in these datasets, and LSTM can model these patterns. The general architecture of an LSTM unfolded cell is depicted in Fig. 10b.

To take advantage of essential features from the input, there must be an option, designed to be used in each state (time step) of the network, to preserve or forget inputs (previous state, bias, and current input data). Alongside the three mentioned gates, there is a memory structure sometimes called the "control" or "candidate" gate (the C_t gate in Fig. 10b) that decides what data to be written to the current cell state.

Based on the LSTM unrolled cell diagram in Fig. 10b, the current state of each cell will be the output of the following equation:

$$Input_t = (W_{I1} \cdot x_t + W_{I2} \cdot h_{t-1} + W_{I3} \cdot c_{t-1} + b_I) \quad (2)$$

$$Forget_t = (W_{F1} \cdot x_t + W_{F2} \cdot h_{t-1} + W_{F3} \cdot c_{t-1} + b_F) \quad (3)$$

$$C_t = Forget_t \cdot C_{t-1} + Input_t \cdot Tanh(W_{C1} \cdot x_t + W_{C2} \cdot h_{t-1} + b_C) \quad (4)$$

$$Output_t = (W_{O1} \cdot x_t + W_{O2} \cdot h_{t-1} + W_{O3} \cdot c_{t-1} + b_O) \quad (5)$$

$$h_t = Output_t \cdot Tanh(C_t) \quad (6)$$

In the above equations, σ and *Tanh* stand for sigmoid and hyperbolic tangent activation functions respectively, W_{ij} is the weight and b_i is the biases respectively for each gate (i) and route (j). The "Tanh" activation function is almost used as a default option in all RNN networks because it outputs any given value, to a number between "−1" and "1" and the relative difference between inputs will be preserved in successive steps of the recurrent flow and this prevents from the gradient exploding phenomena.

5.6 Gated Recurrent Unit Network

Based on the motivation of reducing the complexity and preventing gradient vanishing state, and as a modification to LSTM, the GRU network, was first introduced by Cho et al. (2014) and in this modification, the forget and input gates were combined into a single gate named "update gate". The other gate used in this architecture is the "reset gate" and these two gates control what information to be passed or not. The resulted model is more straightforward than LSTM and hence, is becoming increasingly popular. An unfolded GRU cell is illustrated in Fig. 10c and the corresponding equations are extracted from the information flow as follows:

$$Update_t = (W_{U1} \cdot x_t + W_{U2} \cdot h_{t-1} + b_U) \quad (7)$$

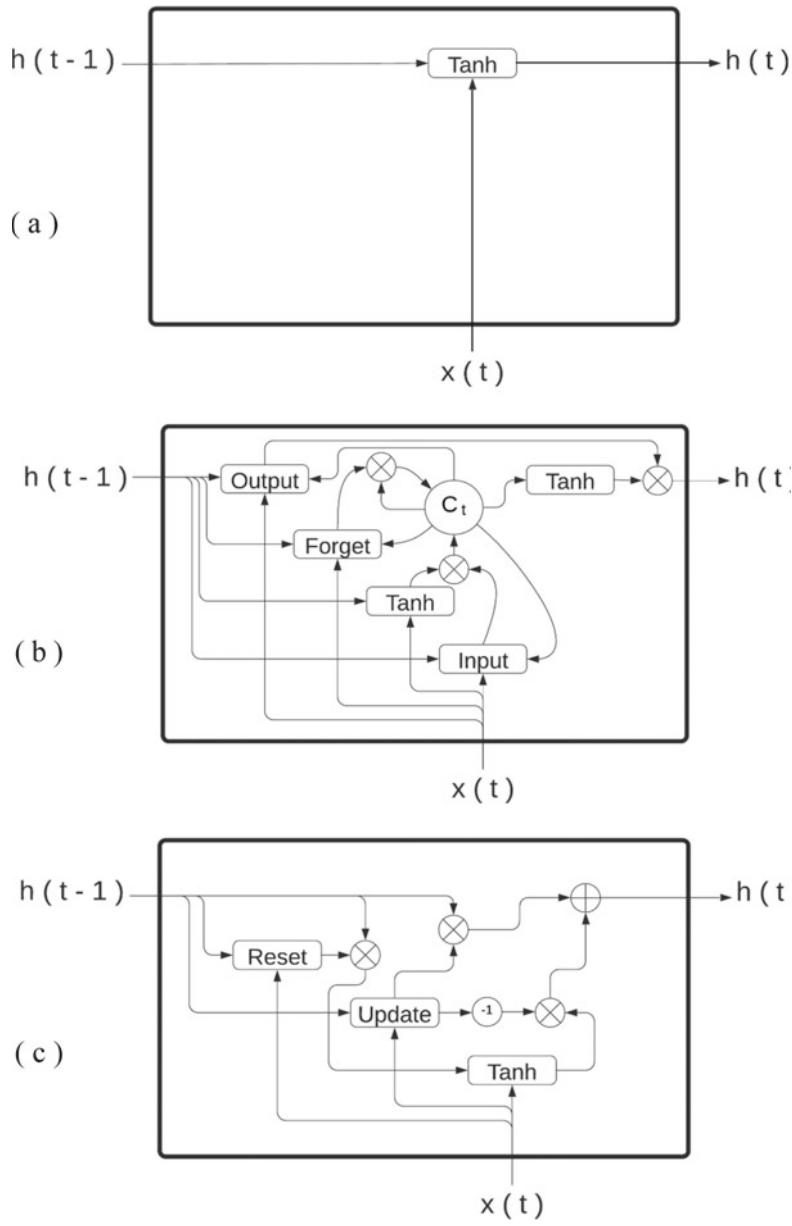


Fig. 10 Schematic illustration of basic unfolded **a** RNN, **b** LSTM, and **c** GRU cells

$$Reset_t = (W_{R1} \cdot x_t + W_{R2} \cdot h_{t-1} + b_R) \quad (8)$$

$$h_t = Update_t \cdot h_{t-1} + (1 - Update_t) \cdot Tanh(W_{h1} \cdot x_t + W_{h2} \cdot (Reset_t * h_{t-1}) + b_h) \quad (9)$$

It is noteworthy that LSTM and GRU have been compared using different conditions and datasets in various studies. In most cases, it is concluded that the latter is superior in terms of performance (Mateus et al. 2021; Yang et al. 2020). However, both are among the first choices in many studies such as prediction and regression problems.

5.7 Application of Long Short Term Memory Network for Extrapolating 2D Sequential Data

In this section, the ability of RNN networks in sequence data prediction is provided in the form of an extrapolation problem. At the very first step, a synthetic seismic gather is produced using a linear earth model. (Any other 2D dataset also could be used instead of the mentioned seismic gather.) For extrapolation, seven rows and 10 columns are used as input data (Figs. 11 and 12). More details are provided below in different parts of the code.

Two modules are installed in the first step in the colab environment to generate a synthetic seismic section and also plot it in the wiggle format:

```
!pip install bruges
!pip install wiggle
```

```
>Collecting bruges
Downloading https://files.pythonhosted.org/packages/67/0c/e4c30c7e14539f4a25e34067440805d0a11d184b947f0d33502e890d077e/bruges-0.4.2.tar.gz (74kB)
|[redacted]| 81kB 3.8MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from bruges) (1.19.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from bruges) (1.4.1)
.
.
.
Successfully built wiggle
Installing collected packages: wiggle
Successfully installed wiggle-0.1.0
```

Now, the required modules are loaded:

```
import numpy as np
import bruges
import wiggle
import matplotlib.pyplot as plt
from IPython.core.pylabtools import figsize
from tensorflow.keras import Sequential, layers, callbacks
from tensorflow.keras.layers import Dense, LSTM, Dropout
```

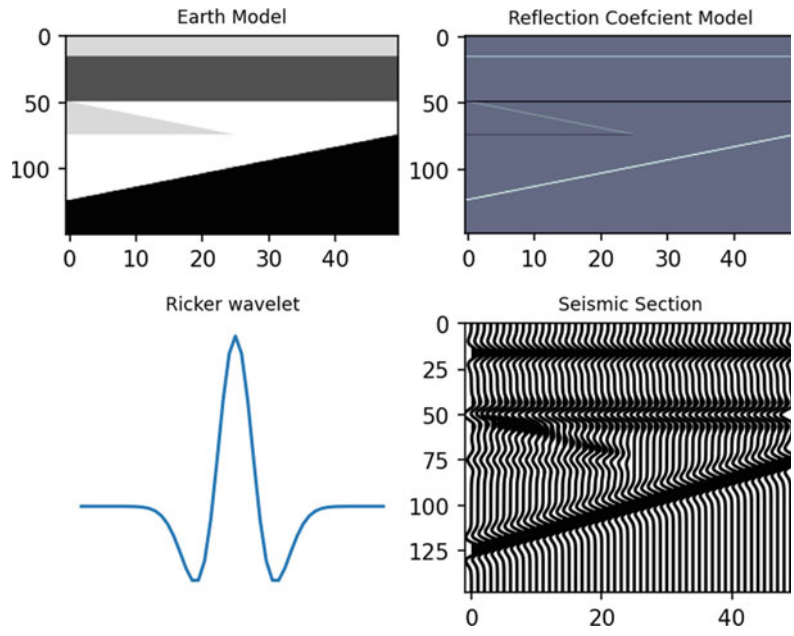


Fig. 11 The earth model and resulted reflection coefficient model, used wavelet and the produced seismic section applied in extrapolation experiment

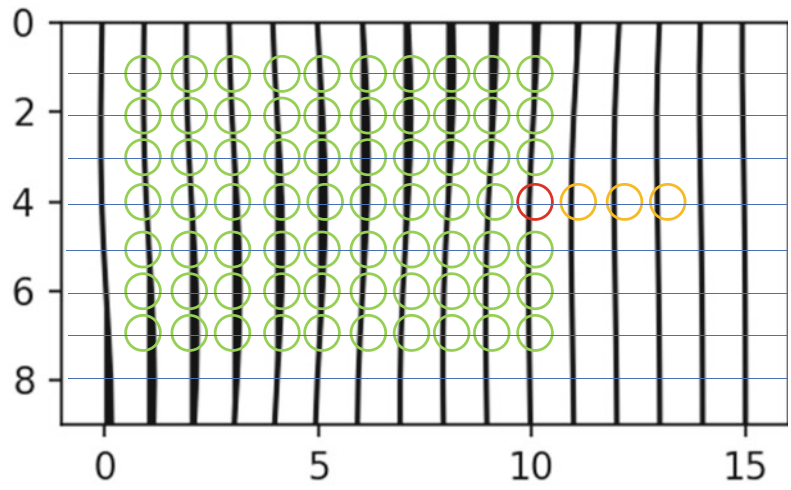


Fig. 12 A zoomed part of a seismic section, the main point (red circle), and the corresponding 69 points (green circles) are used for predicting the values of the three points in subsequent traces (orange circles)

In this part of the code, a synthetic seismic section is generated using a simple linear earth model and convolving the resulting array with a Ricker wavelet. The model (earth and reflection coefficient), wavelet and, resulting seismic section are plotted in Fig. 11:

```
#Construction of the earth model.
length, depth = 50, 150
#Adding two wedges to the model.
w_begin = depth//3
earth_model = np.tri(depth, length,
   -w_begin, dtype=int)
earth_model[:w_begin//3,:] = 1
earth_model[w_begin//3:w_begin,:] = 3
temp = -earth_model[w_begin:w_begin+75]
earth_model[depth-75:depth,:] = 4*earth
   _model[w_begin:w_begin+75,::-1]
#Constructing velocity-density pairs
   to calculate "acoustic impedance"
   will #finally be used in reflection
   coefficient calculation.
rocks = np.array([[800, 1000], [1100,
   1150], [1300, 1300], [1700, 1900],
   [2400, 2450], [2600, 2750]])
earth = rocks[earth_model]
acc_imp = np.apply_along_axis(np.
   product, -1, earth)
ref_coef = (acc_imp[1:,:] - acc_imp
   [:-1,:])/(acc_imp[1:,:] + acc_imp
   [:-1,:])
#Generating a 50 msec Ricker wavelet with
   60 Hz as the center frequency and
   #0.001 time sampling interval.
wvlet = bruges.filters.ricker
   (duration=0.050, dt=0.001, f=60)
synsec = np.apply_along_axis(lambda t:
   np.convolve(t, wvlet, mode='same'),
   arr=ref_coef, axis=0)
#Plotting the results
plt.figure(figsize = [6,5], dpi=150)
ax = plt.subplot(2, 2, 1)
plt.imshow(earth_model, cmap='Greys',
   aspect=0.2)
plt.title('Earth Model',fontsize =
   'small')
ax = plt.subplot(2, 2, 2)
plt.imshow(ref_coef, cmap='bone',
   aspect=0.2)
plt.title('Reflection Coefficient Model',
   fontsize = 'small')
ax = plt.subplot(2, 2, 3, aspect='auto')
plt.plot(wvlet)
plt.title('Ricker wavelet',fontsize =
   'small')
plt.axis('off')
ax = plt.subplot(2, 2, 4)
wiggle.wiggle(synsec,sf=0.55)
plt.title('Seismic Section',fontsize =
   'small')
plt.show()
```

A small part of the data is plotted to show how the consecutive rows and columns are used for data extrapolation (prediction). As shown in the figure, the three rows above and below each point are used. So, considering the central row, seven rows are used for each prediction. Also, considering the backstep parameter, which determines the number of columns (traces), the total length of each input will be 70 points. The center point and the other 69 points are plotted in Fig. 12 with red and green circles, respectively, and the targets are plotted in yellow color:

```
trndata = np.array(synsec)[:,:47]
tstdata = np.array(synsec)[:,47:]
plt.figure(figsize=[3.5,2], dpi=150)
sampledata = np.array(synsec)
  [115:125,0:16]
wiggle.wiggle(sampledata, sf=0.1,
  color='k')
```

As mentioned in the text, one of the possible confusing and challenging parts of using the "Keras" framework is the input structure of the different networks. Here, for preparing training and validation datasets, based on the random selection of the points, the number of back steps, and the number of points to be predicted:

```
#This function, randomly generates
  possible data points for the
  construction
#of the training and validating sub-data.
  The output is simply fed to the
#next function for providing the
  input data
def pointselection(data, trnper, bkstep,
  no_t):
  import numpy as np
  nr, nc = data.shape
  allpoints = []
  [allpoints.append((x, y)) for x in
  range(3,nr-3) for y in range(bkstep,
  nc-no_t)]
  allpoints = np.array(allpoints)
  ntrn = int(np.floor(trnper*0.01*
  allpoints.shape[0]))
  allin = np.random.choice(list(range
  (allpoints.shape[0])),
  allpoints.shape[0], replace=False)
  trnpoints = allpoints[allin[0:ntrn]]
  valpoints = allpoints[allin[ntrn:]]
  return np.array(trnpoints), np.array
  (valpoints)
#each sequence of input that are organizec
  based on seven time samples for
#each trace and an arbitrary backsteps
  (number of traces).
def datafrompoints(data, points, bkstep,
  no_t, target_values='y'):
  import numpy as np
  data = list(data)
  trX = []
  trY = []
  temp= []
  for i in range(points.shape[0]):
    temp3=list(data[points[i,0]-3]
    [points[i,1]-bkstep:points[i,1]]);
    temp2=list(data[points[i,0]-2]
    [points[i,1]-bkstep:points[i,1]]);
    temp1=list(data[points[i,0]-1]
    [points[i,1]-bkstep:points[i,1]]);
    temp0=list(data[points[i,0]]
    [points[i,1]-bkstep:points[i,1]]);
    temp4=list(data[points[i,0]+1]
    [points[i,1]-bkstep:points[i,1]]);
    temp5=list(data[points[i,0]+2]
    [points[i,1]-bkstep:points[i,1]]);
    temp6=list(data[points[i,0]+3]
    [points[i,1]-bkstep:points[i,1]]);
    temp = temp3+temp2+temp1+temp0+
    temp4+temp5+temp6
    trX.append(temp)
    if target_values=='y':
      temp = data[points[i,0]][points
    [i,1]:points[i,1]+no_t]
      trY.append(temp)
  trX = np.array(trX)
  if target_values=='y':
    trY = np.array(trY)
  return trX, trY
```

In this experiment, the last three traces of the generated dataset are considered for testing. From the rest of the data volume, 85% are

considered training and 15% for the validation sets. Training and validation sets are constructed, then their dimensions are changed in the manner which the Keras accepts, i.e., a 3D NumPy array in the order of:

(Number of sequences, backsteps, number of the features)

More details are provided below:

```
#Preparing input and target sets for
   training and validation data and
#calculating the data size.
trpoints, valpoints = pointselection
   (trndata, 85, 10, 3)
trnX, trnY = datafrompoints(trndata,
   trpoints, 10, 3, target_values='y')
valX, valY = datafrompoints(trndata,
   valpoints, 10, 3, target_values='y')
#Data structure for RNNs in the keras:
   number of sequences, time steps,
#elements per timestep
trnX = np.reshape(trnX, (trnX.shape[0],
trnX.shape[1], 1))
trnY = np.reshape(trnY, (trnY.shape[0],
trnY.shape[1], 1))
valX = np.reshape(valX, (valX.shape[0],
valX.shape[1], 1))
valY = np.reshape(valY, (valY.shape[0],
valY.shape[1], 1))
print('For the train data, input (X) and
target (Y), are of sizes', trnX.sha-
pe,'and', trnY.shape, ', respectively.')
print('For the validation data, input
(X) and target (Y), are of sizes',
    valX.shape,'and', valY.shape,
   ', respectively.')
```

>For the train data, inputs (X) and targets (Y), are of sizes (4132, 70, 1) and (4132, 3, 1), respectively.
>For the validation data, inputs (X) and targets (Y), are of sizes (730, 70, 1) and (730, 3, 1), respectively.

The wiggle plot of the first five train vectors, each of which is constructed using the mentioned seven rows of the data, is shown in Fig. 13:

```
wiggle.wiggle( trnX[1:6,:,0].T)
```

The LSTM model applied in this example consists of four LSTM layers. After the first three of the mentioned layers, a "dropout" layer is added to prevent the model from overtraining. The dropout layer randomly sets the input units to "0" with a frequency rate specified in the layer description. For example, in the first dropout layer, "0.5" means that each input of the current layer (on the other hand, the previous layer's output) will be set to zero with a possibility of 50%:

```
#Initializing a Sequential model
model = Sequential()
#Adding the first LSTM layer with the option
   to pass all hidden state #sequence to
   the next layer instead
#of just passing the last one. This
   enhances the predicting procedure.
Also,
#the number of the first (hidden) LSTM
   units is set to 50 units.
model.add(LSTM(units = 50, return_
   sequences = True, input_shape =
   (trnX.shape[1], 1)))
model.add(Dropout(0.50))
model.add(LSTM(units = 25, return_se-
quences = True))
model.add(Dropout(0.25))
model.add(LSTM(units = 10, return_se-
quences = True))
model.add(Dropout(0.25))
model.add(LSTM(units = 5))
#As the network is used to predict the next
three points, the last layer is #consid-
ered a three-unit dense layer.
model.add(Dense(units = 3))
model.compile(optimizer = 'adam', loss =
'mean_squared_error')
model.summary()
```

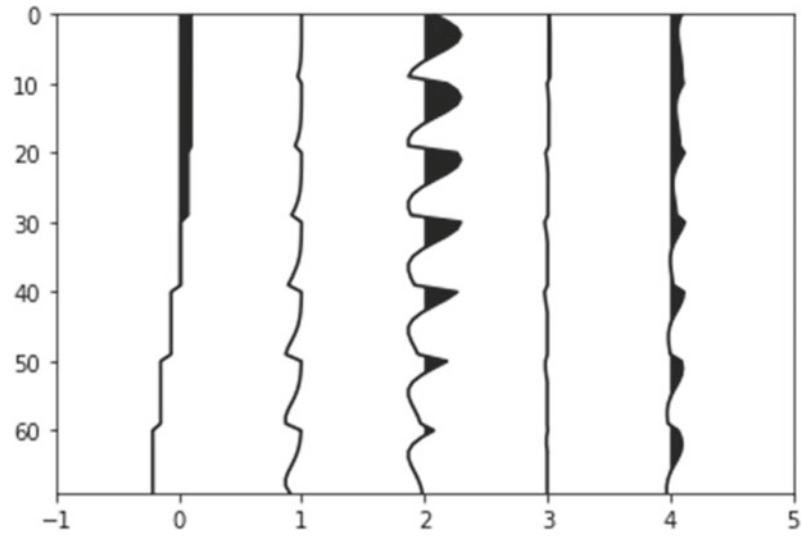


Fig. 13 Five instances of input data used for extrapolating the three consequences points

```
Model: "sequential_1"
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 70, 50)            10,400
dropout_3 (Dropout)          (None, 70, 50)            0
lstm_5 (LSTM)                (None, 70, 25)            7600
dropout_4 (Dropout)          (None, 70, 25)            0
lstm_6 (LSTM)                (None, 70, 10)            1440
dropout_5 (Dropout)          (None, 70, 10)            0
lstm_7 (LSTM)                (None, 5)                 320
dense_1 (Dense)              (None, 3)                 18
=================================================================

Total params: 19,778
Trainable params: 19,778
Non-trainable params: 0
#Training the defined model.
history = model.fit(trnX, trnY, epochs =
20, batch_size = 32 , validation_data=
   (valX, valY))
Epoch 1/20
130/130  [====================]  -  23s
124ms/step  -  loss:  0.0362  -  val_loss:
0.0307
Epoch 2/20
130/130  [====================]  -  16s
119ms/step  -  loss:  0.0302  -  val_loss:
0.0299
Epoch 3/20
130/130  [====================]  -  15s
117ms/step  -  loss:  0.0297  -  val_loss:
0.0291
.
.
.
Epoch 18/20
130/130  [====================]  -  15s
118ms/step  -  loss:  0.0026  -  val_loss:
0.0015
Epoch 19/20
130/130  [====================]  -  15s
117ms/step  -  loss:  0.0026  -  val_loss:
0.0020
Epoch 20/20
130/130  [====================]  -  15s
119ms/step  -  loss:  0.0023  -  val_loss:
0.0014
```

The model's train and validation losses are plotted in Fig. 14:

```
plt.plot(list(range(1,21)),history.
history['loss'])
plt.plot(list(range(1,21)),history.
history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train Loss', 'Validation
  Loss'], loc='upper right', fontsize
  = 'large')
plt.show()
```

In the next step, the predictions for the validation data made by the model are calculated and plotted for the three consecutive traces (Fig. 15). It is noteworthy that these points are chosen randomly, and the plot does not show any coherency. Each point in the plot should be investigated individually. Therefore, the corresponding error is calculated and plotted for ease of performance evaluation (Fig. 15). Also, based on reviewing the sum of absolute errors calculated for the successive traces, the model has performed better in predicting the first step's data, than the two others:

```
predictions = model.predict(valX,
  batch_size=32)
plt.figure(figsize = [7, 5], dpi=150)
ax = plt.subplot(1, 3, 1)
plt.plot(valY[:,0,0],list(range
  (predictions.shape[0])), predictions
[:,0],list(range(predictions.shape
[0])),
    valY[:,0,0]-predictions[:,0],list
(range(predictions.shape[0])))
#Calculating the sum of absolute errors
for each the first to three columns
#of the targets and predictions of the
validation data.
soe1=np.round(sum(abs(valY[:,0,0]-
  predictions[:,0])), decimals=2)
tsoe1='First Trace Information \n'+
  'sum of errors =' + soe1.astype(str)
plt.title(tsoe1,fontsize = 'small')
plt.axis("off")
ax = plt.subplot(1, 3, 2)
plt.plot(valY[:,1,0],list(range
  (predictions.shape[0])),predictions
  [:,1],list(range(predictions.
  shape[0])),
```

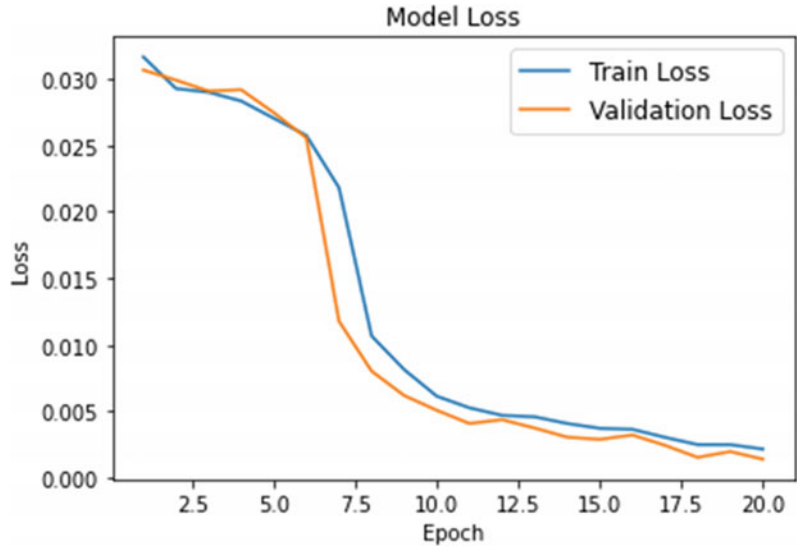


Fig. 14 Plot of the training and validation losses, calculated for each epoch

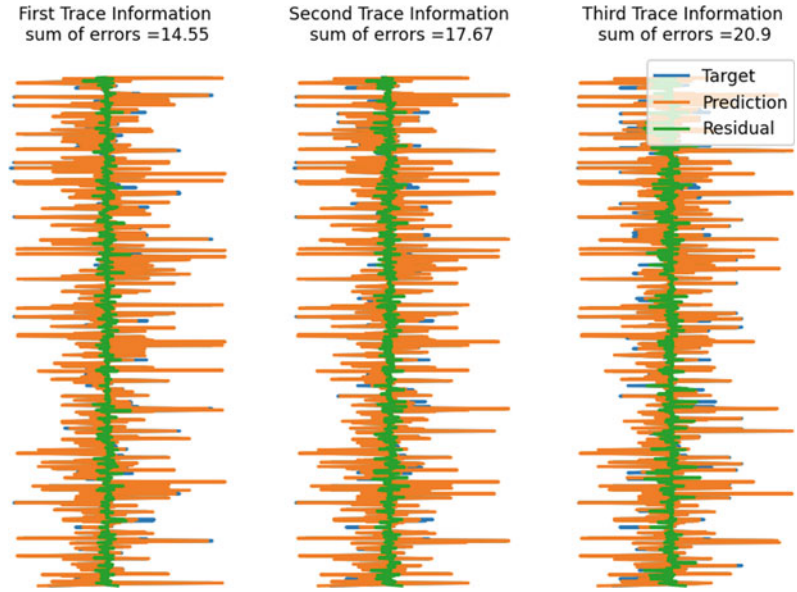


Fig. 15 The predictions for the validation data resulted from generalizing the model for the points selected randomly by the developed functions after grouping them into three subsequent traces

```
      valY[:,1,0]-predictions[:,1],
   list(range(predictions.shape[0])))
soe2 = np.round(sum(abs(valY
   [:,1,0]-predictions[:,1])),
   decimals=2)
tsoe2 = 'Second Trace Information
   \n'+ 'sum of errors =' + soe2.
   astype(str)
plt.title(tsoe2,fontsize = 'small')
plt.axis("off")
ax = plt.subplot(1, 3, 3)
plt.plot(valY[:,2,0],list(range
   (predictions.shape[0])),predictions
   [:,2],list(range(predictions.shape
   [0])),
      valY[:,2,0]-predictions[:,2],
   list(range(predictions.shape[0])))
soe3 = np.round(sum(abs(valY
   [:,2,0]-predictions[:,2])),
   decimals=1)
tsoe3 = 'Third Trace Information \n'+
   'sum of errors =' + soe3.astype(str)
plt.title(tsoe3,fontsize = 'small')
plt.axis("off")
plt.legend(['Target','Predic-
tion','Residual'],loc = 'upper right',
   fontsize = 'small')
plt.show()
```

Finally, the predictions (outputs) for the last three traces of the synthetic section, which were not utilized during the training procedure, are calculated and illustrated in Fig. 16:

```
#Generalizing the trained network to
   traces number 48 to 50, constructing
#the array which contains the specifica-
tion of the points
tstpoints = np.empty([synsec.shape
   [0]-6, 2])
tstpoints[:,0] = list(range(3,synsec.
   shape[0]-3))
tstpoints[:,1] = int(47)
#Extracting the related backstep data
tstX, tstY = datafrompoints(synsec,
   tstpoints.astype(int), 10, 3,
   target_values='n')
tstX = np.reshape(tstX, (tstX.shape[0],
   tstX.shape[1], 1))
#Generalizing the trained network for
    calculating the output
tstpredictions = model.predict(tstX,
   batch_size=32)
#Plotting the predicted traces
plt.figure(figsize = [6, 4], dpi=150)
ax = plt.subplot(1, 3, 1)
ax.invert_yaxis()
```

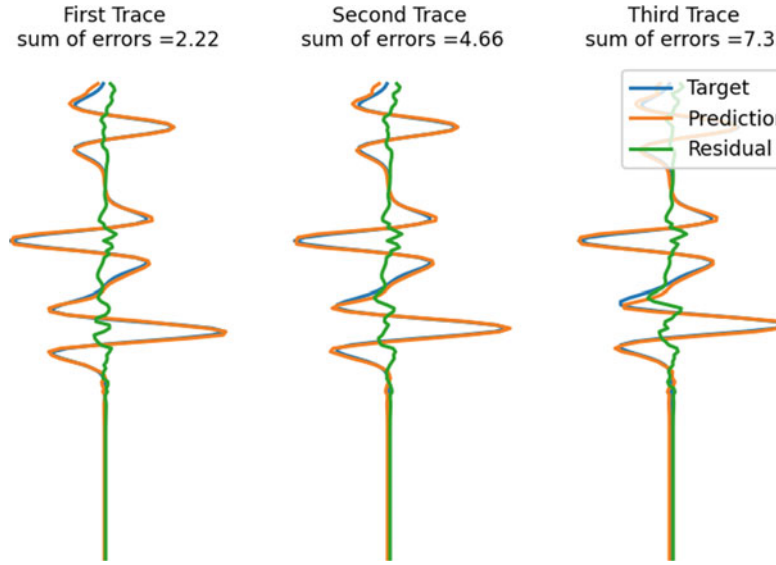


Fig. 16 The results of extrapolating, based on ten subsequent traces (Fig. 13) for the data not involved in training nor validation subsets

```
plt.plot(synsec[list(range(3,synsec.
shape[0]-3)),47].T, list(range
   (tstpredictions.shape[0])))
plt.plot(tstpredictions[:,0], list
   (range(tstpredictions.shape[0])))
plt.plot(synsec[list(range(3,synsec.
shape[0]-3)),47].T -
   tstpredictions[:,0],
      list(range(tstpredictions.shape
[0])))
soe1 = np.round(sum(abs(synsec
   [list(range(3,synsec.shape[0]-
3)),47].T - tstpredictions[:,0])),
   decimals=2)
tsoe1 = 'First Trace \n'+ 'sum of errors ='
   + soe1.astype(str)
plt.title(tsoe1,fontsize = 'small')
plt.axis("off")
ax = plt.subplot(1, 3, 2)
ax.invert_yaxis()
plt.plot(synsec[list(range(3,synsec.
shape[0]-3)),48].T, list(range
   (tstpredictions.shape[0])))
plt.plot(tstpredictions[:,1],
   list(range(tstpredictions.shape
[0])))
plt.plot(synsec[list(range(3,synsec.
shape[0]-3)),48].T -
   tstpredictions[:,1],
      list(range(tstpredictions.
   shape[0])))
soe2 = np.round(sum(abs(synsec
   [list(range(3,synsec.shape[0]-
3)),48].T - tstpredictions[:,0])),
   decimals=2)
tsoe2 = 'Second Trace \n'+ 'sum of errors
   =' + soe2.astype(str)
plt.title(tsoe2,fontsize = 'small')
plt.axis("off")
ax = plt.subplot(1, 3, 3)
ax.invert_yaxis()
plt.plot(synsec[list(range(3,synsec.
   shape[0]-3)),49].T, list(range
   (tstpredictions.shape[0])))
plt.plot(tstpredictions[:,2],
   list(range(tstpredictions.shape
   [0])))
plt.plot(synsec[list(range(3,synsec.
   shape[0]-3)),49].T - tstpredictions
   [:,2],
      list(range(tstpredictions.shape
   [0])))
soe3 = np.round(sum(abs(synsec
   [list(range(3,synsec.shape[0]-
3)),49].T-tstpredictions[:,0])),
   decimals=2)
tsoe3 = 'Third Trace \n'+ 'sum of errors ='
+ soe3.astype(str)
plt.title(tsoe3,fontsize='small')
plt.axis("off")
```

```
plt.legend(['Target','Predic-
tion','Residual'],loc = 'upper right',
   fontsize = 'small')
plt.show()
```

This experiment utilized an LSTM recurrent neural network using an extrapolation or prediction problem. In the developed model, four LSTM layers were considered with 50, 25, 10, and five units, respectively. Also, Dropout layers were used after each LSTM to prevent the model from becoming overtrained. Because the target was to predict three subsequent samples, after mentioned LSTM layers, a dense layer of size, three neurons, was used. The model was trained using the "ADAM" optimizer and "mean_squared_error" loss function. The traces resulting from generalizing the model to the test data support the acceptable performance of the model trained using 20 epochs.

5.8 Unsupervised Deep Learning Methods

Like how most of the process of natural learning takes place in humans, an unsupervised learning method is based on extracting the patterns and structures of the data and generalizing the trained concepts. This goal is achieved through performing operations just to the input data without considering targets, and the interactions are then discovered automatically. The most well-known usage of unsupervised learning is clustering data based on the introduced features. Other applications include data dimension reduction and feature extraction from unlabeled datasets. Two mainly used methods amongst unsupervised DL networks: Auto Encoder Networks, AEN (Hinton and Salakhutdinov 2006) and Generative Adversarial Networks, GAN (Goodfellow et al. 2014) are discussed in the following.

5.9 Unsupervised Auto Encoder Network

The best to describe An AEN is to say it is a mechanism for self-representation. This means, in general, the input is copied (encoded) by the model and generated (decoded) after passing from hidden layers as the output. If a large sequence of hidden layers is used, the model would be considered a deep AEN (Fig. 17). In such a model, each layer will be regarded as new input after being encoded.

Usually, the AENs are not supposed to model the input perfectly and an approximation is enough for the problems such as denoising, data compression, reducing dimensionality, or feature extraction to be solved. On the other hand, the valuable information of an AEN is the encoded information.

Different types of AENs include:

- **Denoising Auto Encoder**: Is used for generating an uncorrupted version of the input data.
- **Sparse Auto Encoder**: Typically learned for another task e.g., classification.
- **Contractive Auto Encoder**: Which is less sensitive to small variation and is suitable for feature extraction problems.
- **Convolutional Auto Encoder**: By using convolutional filters, the model is used for compact representation of the input for any purpose such as classification problems.
- **Variational Auto Encoder**: This model provides more accurate control over modeling and using the latent distribution of the data.
- **Undercomplete Autoencoder**: The purpose of using this network is to capture the main features of the input.

5.10 Attenuating Random Noise in Gravity Data Using Auto Encoder Network

As a common problem, geophysical datasets such as gravity data (Martinez and Li 2016) also suffer from many kinds of contaminated noise,

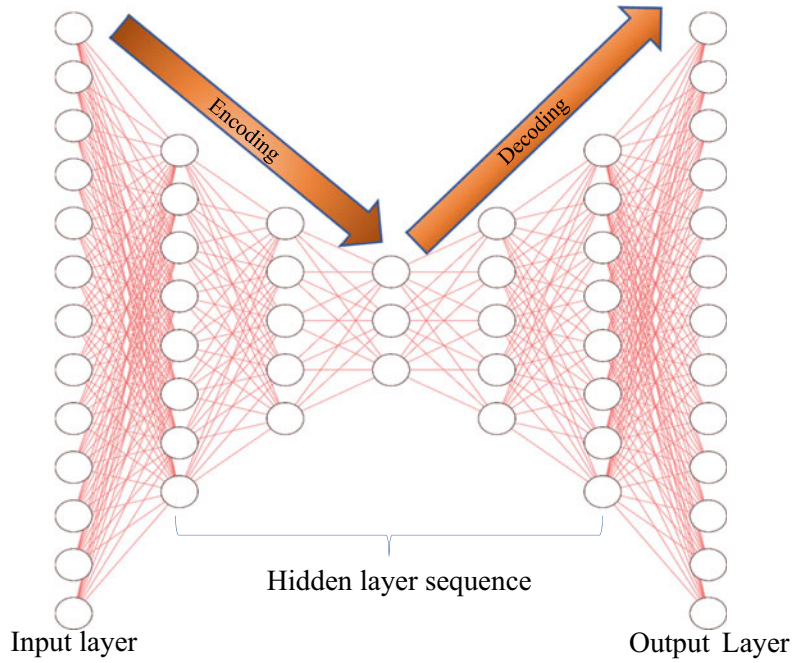


Fig. 17 The general architecture of an AEN model

such as random noise components originating from various sources. Therefore, noise suppression is crucial in the very first steps of geophysical data processing.

In this section, attenuating random noise from synthetic gravity responses of a sphere is using AEN is studied. Generic earth models used here consist of spherical anomalies with different radii and depths buried in a homogeneous medium. The density difference between the anomaly and the surrounding medium, also the amount of additive random noise are generated randomly for each earth model.

The learning data consists of 2000 models 90% of them are dedicated to the train, and the rest are considered validation subsets.

Because the problem here is about denoising, in the AEN model used here, the focus will be on the encoding power of the deep model. In continuation, details about generating earth models, related gravity responses, and the model are provided. Finally, after training the developed model, the results of generalizing to six test models are plotted.

A set consisting of 2006 Random gravity models and their related gravity responses are generated first. The last six models are kept to evaluate the trained autoencoder model and are not included in the training or the validating subsets.

The models are spherical anomalies buried in a homogeneous medium. The random radii of spheres were considered to be chosen from the range of [5, 15] units. The resulting gravitational responses are contaminated with some random noise, and finally, the noisy versions of the responses are achieved:

```
#Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt
#Number of samples
N = 2006
G = 6.6742*10e-11
#Generating random radiuses
R = np.random.randint(5,15, size=(N,1))
#Generating random density difference
   (contrast)
d_rho = np.random.randint(1,100,
   size=(N,1))*0.01
#Generating the random depths of
   the anomalies
Z = np.random.randint(5+R.max(),
```

```
    100, size = (N,1))
#The gravity response of the synthetic
    models are now calculated and by
#generating random arrays, the
#noisy versions of the responses are
    also generated
gra_res = np.empty((N,151), dtype=float)
noisy_gra_res = np.empty((N,151),
    dtype=float)
noise_dev = np.empty((N,1), dtype=float)
x = np.linspace(-100, 100, 151)
for j in range(N):
    cnt=0
    for i in x:
        # Because the calculated gravity
    responses are too small, they are all
        #multiplied by 10e12.
        gra_res[j, cnt] =
        ((4/3)*np.pi*R[j]**3)*G*d_rho[j]*
(Z[j]/(i**2+Z[j]**2)**1.5)*10e10
        cnt +=1
    #Generating and adding noise to the
    gravity response.
    noise_dev[j] = (np.mean(gra_res[j, :])
    *np.random.randint(5,15)*0.01)
    noisy_gra_res[j, :] = gra_res[j,:] +
    np.random.normal(0, noise_dev[j],
    size=(1, gra_res.shape[1]))
```

Here, a randomly selected sample from the earth model dataset and its related gravity response also, its noisy version are plotted (Fig. 18):

```
# Plotting a randomly chosen model and the
corresponded gravity response
rs = np.random.randint(1,N)
plt.figure()
ax1 = plt.subplot(2,1,1)
ax1.set_xlim((-100, 100))
ax1.set_ylim((-Z[rs-1]-2*R[rs-1], 15))
circle = plt.Circle((0, -Z[rs-1]),
   R[rs-1], color='blue')
regt = plt.Rectangle((-100, -Z[rs-1]-2*R
[rs-1]), 200, Z[rs-1]+2*R[rs-1],
   color='yellow')
ax1.add_patch(regt)
ax1.add_patch(circle)
plt.title('Sphere Anomaly in a
   Homogeneous Medium')
ax2 = plt.subplot(2,1,2)
ax2.plot(x,gra_res[rs-1,:])
ax2.set_xlim((-100, 100))
ax2.plot(x,noisy_gra_res[rs-1,:])
plt.title('Gravity Response of the
   Sphere Model')
plt.tight_layout()
plt.show
```

In continuation, the autoencoder model is defined using the Keras API, and then it is compiled and trained. First, the synthetic dataset is partitioned into three subsets, training, validating, and test datasets. The unsupervised autoencoder model is then defined based on the fact that in this experiment, the goal is to denoise the input signal. Hence, the decoding must not be wholly accomplished in signal reconstruction. In doing so, the layers are concentrated in the encoding part of the model, where the size of the successive layers is gradually decreased, and finally, a dense layer the same size as the input is considered for the reconstruction of the input. This decrement forces the model to learn how to translate the inputs in a more simple form or, on the other hand, with less high-frequency random noise.

Considering 100 epochs, the defined model is trained and the related training and validation losses, calculated over each epoch, are plotted in Fig. 19:

```
from keras.models import Sequential
from keras.layers import Dense
#Separating the train, validation and
   test data subsets
val_per = 10
val_idx_start = int(np.ceil((100-
   val_per)*(N-6)*0.01))
trn = noisy_gra_res[0:val_idx_start, :]
val = noisy_gra_res[val_idx_start:
   N-6, :]
tst = noisy_gra_res[N-6:, :]
#Defining the AutoEncoder (AE) Model
AE = Sequential()
AE.add(Dense(units=151,
```

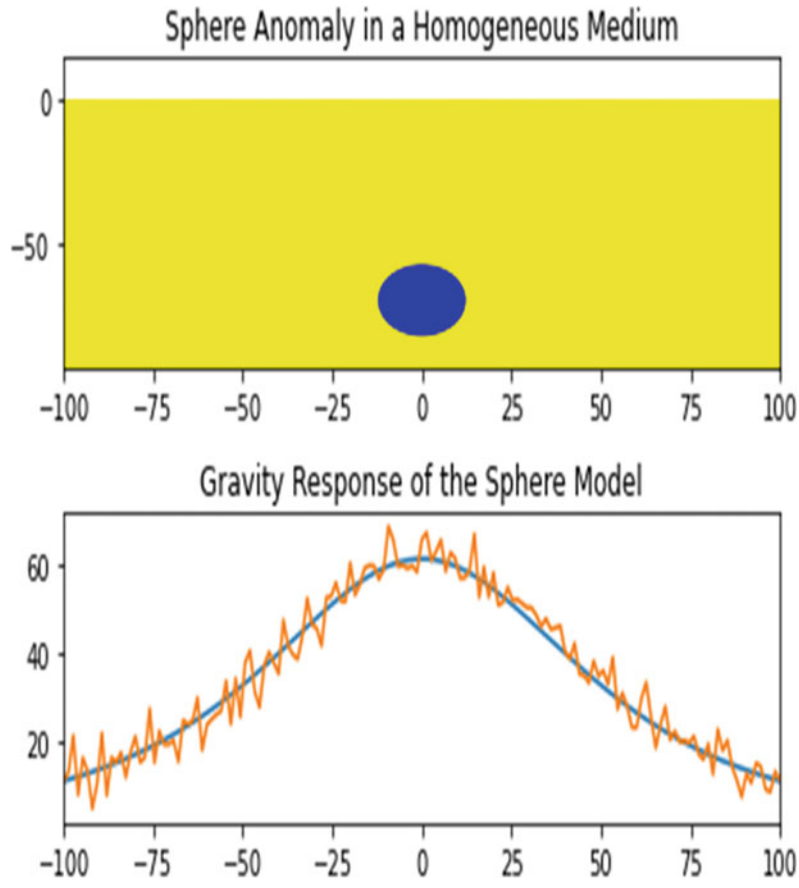


Fig. 18 (Top) A randomly selected model from the set of generated models and (bottom) the related gravity response without (blue) and with additive random noise (red). The Horizontal axis denotes distance in meters, and the vertical axis is 10e−10 Gal

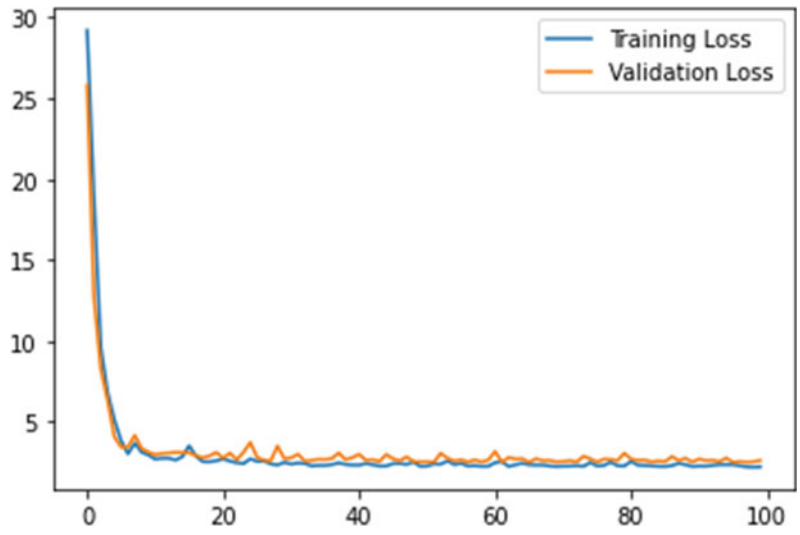


Fig. 19 Plotting the training and validation losses to check the training performance

```
activation='linear'))
AE.add(Dense(512, activation='linear'))
AE.add(Dense(256, activation='linear'))
AE.add(Dense(128, activation='linear'))
AE.add(Dense(64, activation='linear'))
AE.add(Dense(32, activation='linear'))
AE.add(Dense(16, activation='linear'))
AE.add(Dense(5, activation='linear'))
AE.add(Dense(units=151, activation='-
linear'))
#Compiling the model
AE.compile(loss='mae', optimizer='adam')
#Training the compiled model
history = AE.fit(trn, trn, epochs=100,
batch_size=32, validation_data=(val,
val))
>Epoch 1/100
57/57 [==============================] - 1s
10ms/step - loss: 32.902 - val_loss: 25.804
Epoch 2/100
57/57 [==============================] - 0s
7ms/step - loss: 20.3604 - val_loss: 12.721
Epoch 3/100
57/57 [==============================] - 0s
7ms/step - loss: 10.1427 - val_loss: 8.3010
Epoch 4/100
57/57 [==============================] - 0s
7ms/step - loss: 7.7240 - val_loss: 6.3513
.
.
.
57/57 [==============================] - 0s
7ms/step - loss: 2.2074 - val_loss: 2.5532
Epoch 98/100
57/57 [==============================] - 0s
7ms/step - loss: 2.2730 - val_loss: 2.5190
Epoch 99/100
57/57 [==============================] - 0s
7ms/step - loss: 2.2875 - val_loss: 2.5467
Epoch 100/100
57/57 [==============================] - 0s
7ms/step - loss: 2.2127 - val_loss: 2.6150
#Reviewing the defined model
print(AE.summary())
```

```
Model: "sequential_2"
Layer (type)                 Output Shape              Param #
=================================================================
dense_18 (Dense)             (None, 151)               22,952
dense_19 (Dense)             (None, 512)               77,824
dense_20 (Dense)             (None, 256)               131,328
dense_21 (Dense)             (None, 128)               32,896
dense_22 (Dense)             (None, 64)                8256
dense_23 (Dense)             (None, 32)                2080
dense_24 (Dense)             (None, 16)                528
dense_25 (Dense)             (None, 5)                 85
dense_26 (Dense)             (None, 151)               906
=================================================================
```

```
Total params: 276,855
Trainable params: 276,855
Non-trainable params: 0
#Plotting the training and validation
   losses
plt.plot(history.history["loss"],
label="Training Loss")
plt.plot(history.history["val_loss"],
label="Validation Loss")
plt.legend()
```

The last six samples of the generated synthetic dataset were considered for testing the generalization performance of the trained AutoEncoder model. In the plotted charts in Fig. 20, the yellow curves are the ideal outputs (gravity responses without additive random noise). The model's outputs are plotted in blue color. It is worth noting these results are the output of an unsupervised network, and the targets are not fed to the model during the learning procedure.

```
#Generalizing the trained network to test
   data and plotting the results.
tst_out = AE.predict(tst)
tst_target = gra_res[N-6:,:]
#Plotting the results
plt.figure(figsize=(6,5), dpi=150)
plt.subplot(321)
plt.plot(x,tst_target[0],color=
   'yellow', linewidth=2)
plt.plot(x,tst[0], color='red',
   linewidth=1)
plt.plot(x,tst_out[0], color='blue',
   linewidth=1)
plt.subplot(322)
plt.plot(x,tst_target[1],color=
```

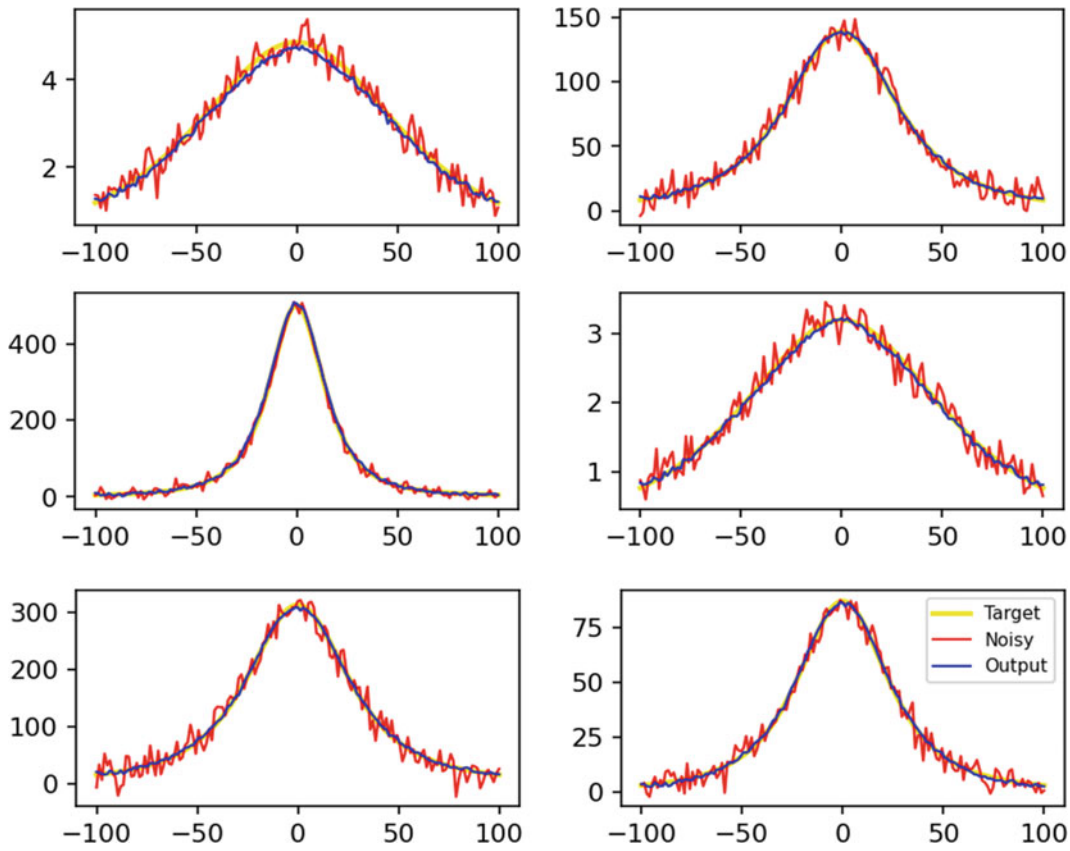


Fig. 20 Results of generalizing the trained model to six test data samples. Here, the yellow curves are the ideal outputs, the red lines are noisy responses, and the blue ones are the denoised responses. The Horizontal axis denotes distance in meters, and the vertical axis is 10e–10 Gal

```
   'yellow', linewidth=2)
plt.plot(x,tst[1], color='red',
   linewidth=1)
plt.plot(x,tst_out[1], color='blue',
   linewidth=1)
plt.subplot(323)
plt.plot(x,tst_target[2],   color='yel-
low',
   linewidth=2)
plt.plot(x,tst[2], color='red',
   linewidth=1)
plt.plot(x,tst_out[2], color='blue',
   linewidth=1)
plt.subplot(324)
plt.plot(x,tst_target[3], color=
   'yellow', linewidth=2)
plt.plot(x,tst[3], color='red',
   linewidth=1)
plt.plot(x,tst_out[3],
   color='blue', linewidth=1)
plt.subplot(325)
plt.plot(x,tst_target[4], color=
   'yellow', linewidth=2)
plt.plot(x,tst[4], color='red',
   linewidth=1)
plt.plot(x,tst_out[4],    color='blue',
linewidth=1)
plt.subplot(326)
plt.plot(x,tst_target[5], color=
   'yellow', linewidth=2)
plt.plot(x,tst[5], color='red',
   linewidth=1)
plt.plot(x,tst_out[5], color='blue',
   linewidth=1)
plt.legend(['Target', 'Noisy',
   'Output'], fontsize=7)
plt.tight_layout()
plt.show
```

In this experiment, an AEN model consists of seven encoding dense layers of sizes 512 to 5 neurons and one decoding layer sized equal to the input signal is used for random noise attenuation purposes. During training the model, the "ADAM" optimizer is used and also "Mean Absolute Error, MAE" is utilized as the loss function. The total epoch is set to 100 and 32 as the batch size.

As an unsupervised algorithm, the capability of the model could be regarded acceptable to be used even in real-world datasets. Although the AENs are individually applied in many problems, they are the fundamental structures used in developing other models, such as the Generative Adversarial Networks discussed in the next section. Another point to be noticed is the plot of losses. For evaluating the model's training performance, the form usually stated to be acceptable is how they are illustrated in this experiment; that is, the corresponding curves for training and validation losses fluctuate around the minimum, near each other. Also, the training loss is supposed to be smaller than of validating; this is the case considered for a well-trained model.

5.10.1 Generative Adversarial Network

Based on the concept of AENs and its capability for new data generation, Generative Adversarial Networks (GAN) are mainly used for producing new samples, recovering the corrupted samples within the dataset, and reforming (reshaping) samples based on the features extracted from the input space.

The basic version of the GAN model consists of two sub-networks: a Generator (G) model that generates new data elements using noisy versions of the samples from the latent space. The latent space refers to the encoded information or learned features from individual inputs to the model.

The output of the G model is fed to the Discriminator (D) model, where, after they are compared with the real data, the model decides whether they are real or fake.

Here, the learning procedure is focused on generating new samples, and the process is continued as long as the generated fake samples are not discriminated against as "fake" ones anymore (Fig. 21). Therefore, one of the main differences between GAN and other models is the dual and contradictory cost functions used in training the model.

In the GAN model, both cost functions are based on the losses resulting from the D model and that is the optimization is toward minimizing

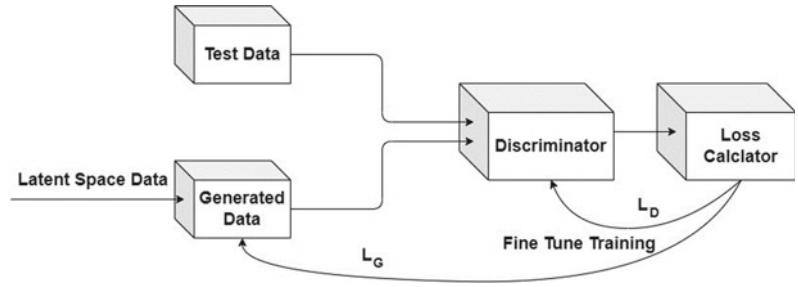


Fig. 21 The general schematic form of the GAN model

the D loss, L_D, and maximizing the G losses, L_G (Fig. 21).

On the hand, meanwhile, the D model tries to be successful in determining all outputs of the G model as fake samples, the G model tries to maximize the loss and this is when, the number of samples has not been discriminated by the D model as fake samples, are maximized. This is called competitive optimization, minimax or a zero-sum game.

The training procedure of a GAN model consists of:

- Selecting a batch of real data samples, x.
- Passing X from sigmoied function in discriminator, D(x).
- Generating a random batch of samples from latent space and feeding the batch to generator, G(z).
- Passing G(z) from dicriminator and sigmoid function, D(G(z)).
- Calculating losses for D and G models:

$$L_D = \log\{\sigma(D(x))\} + \log\{1 - \sigma(D(G(z)))\} \quad (10)$$

$$L_G = \log\{\sigma(D(G(z)))\}\mathrm{D} \quad (11)$$

The σ denotes sigmoid function.

- Back propagating and updating the weights for D and G model respectively.
- All the aforementioned.

Steps are iterated until the optimum weights are achieved.

Different types of GAN models include:

- The Vanilla GAN: The first GAN model introduced by Goodfellow et al. (2014) for image deblurring purposes.
- Deep Convolutional GAN (DCGAN): Which is equipped by deep convolutional layers in both D and G models and is considered as a powerful model for unsupervised learning purposes (Fang et al. 2018).
- Conditional GAN (CGAN): Proposed by Mirza and Osindero (2014), uses additional information such as class labels or data from other sources alongside the input data for both G and D models. This, provides greater control over the output of the G model.
- Info GAN: The model is capable of learning the features hidden in the input data without the need to be labeled. Therefore, the model is known for the interpretable and meaningful representation of the data (Chen et al. 2016).
- Attention GAN: This model was initially introduced by the Microsoft Corporation for producing images based on the texts resulting from Natural Language Processing (NLP) techniques. The model is proven to have excellent performance even when the image is outputted based on a single word as input (Chen et al. 2018).
- Cycle GAN: The model is proposed based on the idea of achieving cyclical changes for a target (Zhu et al. 2017). For example, the

network is trained to output the summer scene by giving a picture of a specific place taken in the winter. The training is performed in such a way that the network itself learns the cyclical changes from the data and not through introducing the elements of the differences. For example, the green leaves are not explicitly taught for summer scenes; instead, the network learns how to properly replace them on the trees in a winter input.

References

Aggarwal CC (2018) Neural networks and deep learning. Springer International Publishing. https://doi.org/10.1007/978-3-319-94463-0

Agostinelli F, Hoffman M, Sadowski P, Baldi P (2014) Learning activation functions to improve deep neural networks

Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili A, Duan Y, Al-Shamma O, Santamaría J, Fadhel MA, Al-Amidie M, Farhan L (2021) Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8(1). https://doi.org/10.1186/s40537-021-00444-8

Bhandarkar T, Satish VKN, Sridhar S, Sivakumar R, Ghosh S (2019) Earthquake trend prediction using long short-term memory RNN. Int J Electr Comput Eng (IJECE) 9(2):1304. https://doi.org/10.11591/ijece.v9i2.pp1304-1312

Bisong E (2019) Google Colaboratory. In: Building machine learning and deep learning models on google cloud platform. Apress, pp 59–64. https://doi.org/10.1007/978-1-4842-4470-8_7

Cai Y, Shyu M-L, Tu Y-X, Teng Y-T, Hu X-X (2019) Anomaly detection of earthquake precursor data using long short-term memory networks. Appl Geophys 16(3):257–266. https://doi.org/10.1007/s11770-019-0774-1

Chen X, Duan Y, Houthooft R, Schulman J, Sutskever I, Abbeel P (2016) InfoGAN: interpretable representation learning by information maximizing generative adversarial nets. http://arxiv.org/abs/1606.03657

Chen X, Xu C, Yang X, Tao D (2018) Attention-GAN for object transfiguration in wild images

Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation

Colom M, Morel J-M (2019) Full-spectrum denoising of high-SNR hyperspectral images. J Opt Soc Am A 36 (3):450. https://doi.org/10.1364/JOSAA.36.000450

Dey R, Salem FM (2017) Gate-variants of gated recurrent unit (GRU) neural networks. In: 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS), pp 1597–1600. https://doi.org/10.1109/MWSCAS.2017.8053243

Fang W, Zhang F, Sheng VS, Ding Y (2018) A method for improving CNN-based image recognition using DCGAN. Comput Mater Contin 57(1):167–178. https://doi.org/10.32604/cmc.2018.02356

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial networks. Adv Neural Inf Process Syst 3(11). https://doi.org/10.1145/3422622

Gulli A, Pal S (2017) Deep learning with Keras. Packt Publishing Ltd

Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507. https://doi.org/10.1126/science.1127647

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, Jackel LD (1989) Backpropagation applied to handwritten zip code recognition. Neural Comput 1(4):541–551. https://doi.org/10.1162/neco.1989.1.4.541

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324. https://doi.org/10.1109/5.726791

Liu L, Sun X-K (2020) Volcanic ash cloud diffusion from remote sensing image using LSTM-CA method. IEEE Access 8:54681–54690. https://doi.org/10.1109/ACCESS.2020.2981368

Liu L, Liu Y, Luo Y (2020) RNN-based dispersion inversion using train-induced signals. SEG Tech Program Expand Abstr 2020:3437–3441. https://doi.org/10.1190/segam2020-3427517.1

Martinez C, Li Y (2016) Denoising of gravity gradient data using an equivalent source technique. Geophysics 81(4):G67–G79. https://doi.org/10.1190/geo2015-0379.1

Mateus BC, Mendes M, Farinha JT, Assis R, Cardoso AM (2021) Comparing LSTM and GRU models to predict the condition of a pulp paper press. Energies 14 (21):6958. https://doi.org/10.3390/en14216958

Mirza M, Osindero S (2014) Conditional generative adversarial nets

Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Signal Process 45 (11):2673–2681. https://doi.org/10.1109/78.650093

Sultana F, Sufian A, Dutta P (2019) Advancements in image classification using convolutional neural network. https://doi.org/10.1109/ICRCICN.2018.8718718

Yang S, Yu X, Zhou Y (2020) LSTM and GRU neural network performance comparison study: taking yelp review dataset as an example. In: 2020 international workshop on electronic communication and artificial intelligence (IWECAI), pp 98–101. https://doi.org/10.1109/IWECAI50956.2020.00027

Zhang A, Lipton ZC, Li M, Smola AJ (2021) Dive into deep learning

Zhu J-Y, Park T, Isola P, Efros AA (2017) Unpaired image-to-image translation using cycle-consistent adversarial networks. In: 2017 IEEE international conference on computer vision (ICCV), pp 2242–2251. https://doi.org/10.1109/ICCV.2017.244

Deep Learning: Applications in Seismology and Volcanology

Abstract

In this chapter, after presenting a list of the DL applications with brief descriptions, various advanced applications of deep learning in the fields of seismology and volcanology are presented. The examples are arranged in two separate parts, one for seismological and the other for volcanological aspects.

1 Introduction

In Chapter "Deep Learning: The Concepts", some basic and general concepts of deep learning were discussed with an emphasis on proposing a descriptive starting point which will be followed by some advanced applications in the fields of seismology and volcanology in this chapter.

2 Applications of Deep Learning in Seismology

It is no exaggeration to say that different deep learning models have been used in almost all significant problems of seismology. Starting from the occurrence of an earthquake, the fault's mechanism analysis, or even sooner than that, prediction of earthquakes, to later related studies such as, processing earthquake waveforms, locating events, energy release, hazard analysis and so on, all have been investigated by many researchers using deep learning methods. As, the main aim of this chapter is to review such studies, At the very first step, a subject-based table is provided for the respected reader that summarizes some valuable published research in each field. Table 1 is organized in such a way that each major topic of earthquake seismology corresponds to a few related studies. Afterward, starred items have been discussed separately in detail.

2.1 Long-Range-Short-Term Earthquake Prediction Using CNN-BiLSTM-AM Model

Many research articles are presented regarding earthquake prediction problems using three main approaches: mathematical, precursors, ML methods. Considering the number of recent publications related to the last-mentioned domain, one may conclude that earthquake prediction utilizing ML methods is a hotspot field in seismological studies. As a quick introduction to the topic, Fig. 1 shows a diagram of the general form of the earthquake prediction problem and the application of machine learning in the approach. Banna et al. (2020) have performed valuable research on the context of using ML models in earthquake prediction. They provided in-depth statistical studies; the respected readers could refer to, in case of interest.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

A. Hajian et al., *Intelligent Methods with Applications in Volcanology and Seismology*, Advances in Volcanology, https://doi.org/10.1007/978-3-031-15432-4_5

Table 1 Brief review of some examples using DL models in seismology

Article no.	1	By weighting and combining simple feature extraction (CNN model output) as implicit features and the seismological indicators as explicit ones, a new DL model named DLEP has been introduced and trained based on the authors' proposed loss function. The Multi-Area Under Curve (MAUC) index, calculated for eight different data sets, has indicated the method's excellence in accurately predicting earthquakes
Main subject	Earthquake prediction	
Article title	DLEP: a deep learning model for earthquake prediction (Li et al. 2020)	
Used model	CNN	
Article no.	2	Here, alongside developing a method for predicting the biggest upcoming earthquake in the next week period, a comparison has also been performed over SVM, DT, SNN and DL. Furthermore, a new parameter called "fault density" has been introduced in the paper and several quantitative indexes have been used for the evaluation of using this parameter for earthquake prediction studies. Different combinations of 16 parameters and classes of earthquake magnitudes have been used for performance evaluation, and finally, it has been concluded that, DL and SVM models have had promising performance versus other models
Main subject	Earthquake prediction	
Article title	Spatiotemporally explicit earthquake prediction using deep neural network (Yousefzadeh et al. 2021)	
Used model	DNN (successive combination of dense and dropout layers)	
Article no.	3	Using more than 600 million data samples in the research, the time to the next fault failure has been investigated, where, the raw data are fed to a CNN network for feature extraction, before an LSTM network for the prediction purpose. Based on the results of the R-squared value computed over the outputs, the proposed model delivers "satisfactory predicted results" in contrast to the conventional catalogue-based studies (Naive forecasting is used here) especially in the case of failure impending
Main subject	Earthquake prediction	
Article title	Deep and confident prediction for a laboratory earthquake (Pu et al. 2021)	
Used model	CNN for feature extraction and LSTM for prediction	
Article no.	4*	The proposed method in the research is based on CNN-BiLSTM-AM model, where, "AM" stands for the "Attention-Mechanism" layer (the layer is used for emphasizing influential information). The primary purpose of the research is to forecast the maximum magnitude and the number of earthquakes in the next one month and the study has been performed on mainland China where the region is divided into nine rectangle-shaped subregions for more precise location prediction. Seven other models, including CNN-BiLSTM model, have been used for performance comparison and the superiority of the method has been concluded regarding three different quantities (RMS, MAE, and R-squared)
Main subject	Earthquake prediction	
Article title	A CNN-BiLSTM model with attention mechanism for earthquake prediction (Kavianpour et al. 2021)	
Used model	CNN and bidirectional LSTM (BiLSTM)	

(continued)

Table 1 (continued)

Article no.	5*	Based on training 787,320 synthetic samples and proposing a novel DL model called FMNet, the focal mechanism of four earthquakes has been solved on a millisecond scale using a single CPU computing system. Three-component simulated waveforms are generated based on the variations of the strike, dip, and rake angles for 16 stations. The input of the DL model is two-dimensional arrays consisting of three-component data of the mentioned 16 stations, and the output is three Gaussian probability distributions corresponding to the three angles of the focal mechanism. Using Kagan angle distribution and applying the trained model for a set of (unseen) 1000 synthetic samples, it has been concluded that the model can predict focal mechanisms with an accuracy of 97.8% with errors within 20° range
Main subject	Focal mechanism	
Article title	Real-time determination of earthquake focal mechanism via deep learning (Kuang et al. 2021)	
Used model	Fully convolutional network (FCN)	
Article no.	6	About 110,000 relatively shallow microearthquakes (with depths smaller than 20 km), which occurred in Japanese islands, are studied for the focal mechanism approach. The proposed method is based on a network trained with an extensive database consisting of moderate to large earthquakes and also re-trained by microearthquakes for fine-tuning consideration. In the applied architecture, a combination of 1D convolutional, batch normalization and max pooling layers that are followed by two flatten layers have been used and the model is trained for outputting downward and upward labels. Also, a data augmentation has been applied using a flipped version of the input data. The authors provide a detailed discussion about the results and their usage, and it has been concluded that the method has almost determined the focal mechanism correctly for all test inputs
Main subject	Focal mechanism	
Article title	Focal mechanisms of small earthquakes beneath the Japanese islands based on first-motion polarities picked using deep learning (Uchide 2020)	
Used model	CNN	
Article no.	7	A real-time method based on spectrograms calculated for local earthquakes is proposed. The output of short-term Fourier transforms of the 3-component waveforms (4 s, 100 Hz time series) are fed to a CNN model, Seismo-Performer, that has been specifically modified for phase picking approach based on "transformer architecture". The training dataset consists of 4.5 million 3-component seismograms recorded in Southern California. As the test datasets, the earthquake records of three different regions have been selected and combined in two different scenarios. Finally, it has been concluded that although the proposed method reserves the performance of the classification, it is computationally more efficient than the other three compared methods, all with DL architecture
Main subject	Phase picking	
Article title	The seismo-performer: a novel machine learning approach for general and efficient seismic phase recognition from local earthquakes in real time (Stepnov et al. 2021)	
Used model	CNN	

(continued)

Table 1 (continued)

Article no.	8	The proposed multitask model, consists of 1D convolutions, bi/uni directional LSTM, Network-in-Network, residual connections, transformer, feed-forward and self-attentive layers. The output of the trained model is supposed to analyze the inputs (earthquake stream/signals) for the existence of an earthquake and P and S phase timings. The training dataset consists of one million earthquakes plus 300 thousand noise signals published specifically for AI studies. The dataset has been split into training (85%), validation (5%), and test (10%) sets. The trained model has been successfully applied to a stream of five weeks of data, recorded during the 2000, Tottori earthquakes in Japan, and authors have stated that the model has been able to detect and locate two times more earthquakes using only less than 1/3 of seismic stations. Also, the model has picked P and S phases with precision close to manual picks performed by human analysts
Main subject	Phase picking	
Article title	Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking (Mousavi et al. 2020)	
Used model	CNN, BiLSTM and LSTM	
Article no.	9*	The developed model in the research, Picknet, has been proposed based on the motivation to perform the phase picking procedure rapidly and precisely for local earthquakes. Picknet architecture is defined in a functional manner and several side layers are deployed on the basis of the main CNN structure. The training dataset consists of about 740,000 manually picked samples, which, are recorded in 782 stations in Japan. The trained model has been evaluated using 300 earthquakes, and the results show that the method, has picked the P and S phase, about ten times more accurately than of phases reported by the Japan Meteorological Agency
Main subject	Phase picking	
Article title	Deep learning for picking seismic arrival times (Wang et al. 2019)	
Used model	(Functional) CNN	
Article no.	10	Using 19,263 100 Hz three-component processed accelerograms recorded by the K-Net (Japan) and the proposed (deep) CNN model named DCNN-M, a method is presented mainly for magnitude calculation to be used in earthquake early warning systems. The model is designed using four groups of convolutional, pooling and batch normalization layers followed by three fully connected layers. A 20% portion of mentioned is used as testing data and performance evaluation. It is concluded that, using the proposed method, not only the standard deviation of the magnitude estimation error reduced significantly in contrast with the τc and Pd methods, it is not dependent on epicentral distance
Main subject	Magnitude calculation	
Article title	Magnitude estimation for earthquake early warning using a deep convolutional neural network (Zhu et al. 2021)	
Used model	CNN	

(continued)

Table 1 (continued)

Article no.	11*	An end-to-end, fast and reliable method based on a composition of CNN and BiLSTM is presented in the research. Since the proposed model (MagNet) is not sensitive to data normalization, it is fed by about 300,000 raw earthquake waveforms just after performing a bandpass filter (1.0–40.0 Hz). The input dataset has been separated to 70% training, 10% validating and 20% testing subsets and based on the revealed results, it is stated that the method is capable of calculating magnitude based on single-station waveforms with an average error close to zero and standard deviation about 0.2 even without any instrumental response correction processing step
Main subject	Magnitude calculation	
Article title	A machine-learning approach for earthquake magnitude estimation (Mousavi and Beroza 2020)	
Used model	CNN, BiLSTM	
Article no	12	The method incorporates computed theoretical seismograms alongside three-dimensional CNN for the determination of earthquake location parameters (time and hypocentral parameters). Inputs to the model are spatial images resulting from the propagation of (theoretical) seismic waves that are simulated using a three-dimensional earth model. The architecture of the three-dimensional CNN model consists of successive 3d convolution and pooling layers followed by a fully connected layer. As input data, 600,000 images of size 32 * 32 corresponding to 2632 earthquakes were used. The method has been evaluated using 159 earthquakes that occurred in the Hakone region, Japan, between 2015 and 2019. Finally, it is acknowledged that the technique is capable of determining earthquake location and the method can potentially serve in monitoring earthquake activities
Main subject	Earthquake locating	
Article title	Application of deep learning-based neural networks using theoretical seismograms as training data for locating earthquakes in the Hakone volcanic region, Japan (Sugiyama et al. 2021)	
Used model	Three-dimensional CNN	
Article no.	13*	A scalable and fast method is presented that determines earthquake location using single station waveforms. The developed model is, ConvNetQuake, is designed using eight convolutional, one flatten and one fully connected layer. The model is trained and evaluated by two combined sets of data. The first data is related to 2918 labeled earthquakes that occurred between 15 February 2014 and 16 November 2016 in Oklahama, USA. The second one which includes 831,111 noise data, is used to fill the gaps between cataloged events. An augmentation procedure has been also used in order to balance the event-noise ratio. After all, the dataset is split into training (90%) and test (10%) subsets. And the model is trained for two targets: detection and location of earthquakes. True events have been correctly detected by the model entirely and the locating accuracy was 74.5%, which is achieved in less than two minutes for all events. Detailed and comprehensive comparison as well as the discussion is provided by the authors
Main subject	Earthquake locating	
Article title	Convolutional neural network for earthquake detection and location (Perol et al. 2018)	
Used model	CNN	

(continued)

Table 1 (continued)

Article no.	14	Two different architecture is proposed for calculating P wave arrival time, epicentral distance and station-to-epicenter back-azimuth. Furthermore, a comprehensive study related to uncertainties has also been provided in the research using the Bayesian framework (quantities are converted to probability distributions). The first model, named "dist-PT network", is a one-dimensional residual TCN network that is fed by four signals, three of which are waveforms (three-component) and the fourth is a binary time series which, "1" valued samples, corresponding to the most essential part of the waveforms. The architecture uses causal and dilated convolution operators where the dilation factor increases exponentially with a power of two (due to the purpose of achieving multi-scale learning). The second developed model, BAZ network, has a functional one-dimensional convolutional structure designed for learning based on two inputs: three-component waveforms and covariance, eigenvalues and the eigenvectors matrix calculated from the waveforms. Finally, and after passing from feature extraction layers, two fully connected layers receive inputs and provide the outputs. A subset of 80% as training and 20% for testing from about 150,000 waveforms are used for training and evaluating the models. Results reveal the absolute mean error for prediction of epicentral distance is 0.23, for P phase travel time is 0.03 s and for back-azimuth is 1.0°. It is concluded that the presented approach could be used even for sparsely recorded earthquakes as a fast method for source characterization
Main subject	Earthquake locating	
Article title	Bayesian-deep-learning estimation of earthquake location from single-station observations (Mousavi and Beroza 2019)	
Used model	CNN, TCN (temporal convolutional network)	
Article no.	15	By using the existing one-dimensional shear wave velocity models, the theoretical phase and group velocities of Rayleigh waves for different periods are computed. The resulting dispersion curves are then, converted to energy images using a Gaussian function. Further, images are fed to CNN model and the trained model is generalized for all grid points of the region of study (continental China and also southern California) for the shear wave velocity model. For the continental China case, 6803 pairs of phase-group velocity image were generated and used as input to the model and after training the model, 3260 evaluation points were used. The resulting v_s maps are provided by the authors for different depths
Main subject	Subsurface studies	
Article title	Using deep learning to derive shear-wave velocity models from surface-wave dispersion data (Hu et al. 2020)	
Used model	CNN	

(continued)

Table 1 (continued)

Article no.	16	A deep-stacked sparse auto-encoders (sSAE) model is introduced and using 12 instances of the model, the crustal thickness (target) is calculated using phase velocity of the Rayleigh surface waves (input). The mode stated to learn high-level features from the raw input data effectively. In general, the proposed method is an inverse problem that is designed using bellow below starting from an earth model, generating the theoretical phase velocity of Rayleigh waves based on the model, training model with the noisy version of the resulting phase velocity to target the crustal thickness and finally using the observed data instances, the best model is selected. Six layered sSAE models are trained and evaluated using, 500,000 data samples where, 76% is allocated to training and the rest for evaluation. It is concluded that the method can be applied for studies like crustal thickness with high resolution
Main subject	Subsurface studies	
Article title	Inverting Rayleigh surface wave velocities for crustal thickness in eastern Tibet and the western Yangtze craton based on deep learning neural networks (Cheng et al. 2019)	
Used model	Auto encoder	
Article no.	17*	Based on the novel approach of the capsule networks, CapsNet, a method for the classification of microseismic events is presented based on the motivation to develop an automated method that is less dependent on large training datasets. The matrix of the features consists of 21 commonly used features that are determined in both time and frequency domains. Since each microseismic record is divided into 33 windows, each input is of size 21 * 33. Following common layers used in a CNN model, units of primary capsule and digit capsule are applied. Five different data, containing different numbers of events (from 400 to 2000), are used for the preparation of training and validating (20%) sets, and another dataset, containing 3200 microseismic records, is used for the evaluation of the model. Based on performing five different training processes, and resulting from studying different quantitative accuracy indexes, it is concluded that the proposed method is superior to compared methods even with smaller training databases
Main subject	Ground motion	
Article title	Microseismic records classification using capsule network with limited training samples in underground mining (Peng et al. 2020)	
Used model	Deep capsule neural network	
Article no.	18	Strong ground motion parameters, PGA, PGV and PGD, have been investigated using the developed DNN model. Values of moment magnitude, Rake angle, the closest distance to fault rupture plane and Vs30 (average shear wave velocity for the first 30 m) are considered as inputs for 12,556 records, cataloged by Pacific Earthquake Engineering Research Center and reported as NGA-West2 database. The data is portioned to 80 and 20% as training and validation sets. Four evaluation indices
Main subject	Ground motion	
Article title	Predicting the principal strong ground motion parameters: a deep learning approach (Derakhshani and Foruzan 2019)	
Used model	DNN	

(continued)

Table 1 (continued)

		including RMSE are used for performance study and based on the performed comparisons, it is concluded that the presented method gives reliable estimates of the strong ground motion parameters and could be used in earthquake engineering applications
Article no.	19	Aiming to synthesize 3-Component accelerograms, a conditional Wasserstein GAN model is proposed and trained using 260,764 strong-motion records published by Japanese seismograph networks (K-NET and KiK-Net). Conditional variables for the generation of the models are: event-station distance, earthquake magnitude and vs30 at the recording stations. The training dataset includes 20-s windowed time series that start 2.5 s before the P phase onset resampled at 20 Hz sampling frequency. Generated seismograms are studied and evaluated using different aspects and it is concluded that: the outputted seismograms display clear P and S-phases, the synthesized PGA estimates are consistent with observations, the trained generator network can be used for PGA interpolation even for the regions where no recordings exist
Main subject	Ground motion	
Article title	Data-driven accelerogram synthesis using deep generative models (Florez et al. 2020)	
Used model	Conditional wasserstein GAN	
Article no.	20*	The purpose of the paper is to seek a method for intensity measurements, that is, fast and reliable enough to be used in real-time data streams analysis. A total number of 700 earthquakes and 1037 noise-related events recorded in 39 stations of the IV network (Italy) were used for training (80%) and validating (20%) a CNN model. Inputs for each event (earthquake), consists of the normalized version of 10 s, three-component acceleration waveforms for all mentioned 39 records, and since, the maximum amplitude of the records physically relates to the targets (strong motion parameters), the normalization factor is saved and is fed to the last fully connected layer of the model. Based on studying the results, it isoncludeed that the proposed method can accurately predict Earthquake intensities at stations even for the points that are far from the epicenter and for which even have not yet recorded the maximum ground motion. Also, it is found that the method could provide estimates of ground motions within 15–20 s after earthquake origin time. Depending on various setup elements
Main subject	Ground motion	
Article title	Rapid prediction of earthquake ground shaking intensity using raw waveform data and a convolutional neural network (Jozinović et al. 2021)	
Used model	CNN	

*Afterward, starred items have been discussed separately in detail

According to the conventional definitions in the field of earthquake prediction studies, a long-range-short-term prediction refers to as a prediction that is reported for a territorial dimension comparable to 100 times of the rupture length (L) of a target earthquake, and in a temporal period, about one to several weeks (Kossobokov et al. 2002).

As briefly described in row 4 of Table 1, Kavianpour et al. (2021) introduced a novel method for earthquake prediction based on the dataset provided from 11,442 earthquakes with M > 3.5 and occurring time between January 15, 1966, to May 22, 2021, in the China mainland. The study area has been divided into nine subregions, as illustrated in Fig. 2.

The DL model has been trained and generalized over catalog information such as latitude and longitude, time of earthquakes, magnitude, depth of earthquakes, etc.

The CNN-BiLSTM model has been used for promising performance in spatiotemporal feature extraction and AM layer has been deployed for emphasizing the essential characteristics of the data that have a strong correlation with positive results. The sequential architecture of the proposed model is illustrated in Fig. 3. At the input block, the "ZOH" stands for Zero Order Hold technique, which is originally a signal reconstruction method and is used to handle the problem of missing or zero-valued features in the dataset. Each of the four successive one-dimensional convolutional blocks also comprises rectified linear unit (ReLU), Max-pooling, and Batch Normalization (BN) layers.

In the sequence learning block, two BiLSTM layers are deployed. As one the most bolded concepts of the paper, before the prediction layer, the output of the BiLSTM is fed to AM layer that, by performing weight allocation, determines the most effective information by distributing higher weights. The functionality of the AM layer is based on three steps: similarity calculation and normalization and "attention value" calculation. The similarity or correlation score, st, is calculated from Eq. 1 for each feature as:

$$s_t = \tanh(W_h h_t + b_h) \tag{1}$$

where W_h, h_t and b_h denote weight, input vector and bias respectively. The similarity score is then normalized over all features, and a softmax function is performed on the resulting values:

$$a_t = \frac{\exp(s_t)}{\sum_t \exp(s_t)} \tag{2}$$

Finally using the weighted summation formula, the output of the AM layer will be calculated as:

$$am = \sum_t a_t h_t \tag{3}$$

The detailed specification of each layer of the model is provided in Table 2. Since the input data is one-dimensional, the convolutional filters used in the "Conv" layers are also considered one-dimensional (e.g. the "Conv1" has been described as a 16 * 5 * 1 layer which means, it is composed of 16 kernels of size five by one).

As mentioned before, there were two main goals studied in the research: predicting the number and magnitudes of earthquakes; therefore, two scenarios have been deployed separately. Due to the time range considered for predictions and the total time range covering earthquakes, the number of inputs, will be limited to 665 samples and a portion of 80% for training and 20% for validation has been used by the authors. Some other critical hyperparameters of the training procedure are summarized in Table 3.

The results of the paper are indicative of the superiority of the proposed method in predicting the number and magnitudes of earthquakes in comparison with other methods. Plots of real data alongside the output of other methods with acceptable performance for respectively subregions 5 and 1, are depicted in Fig. 4

Obviously, with a considerable difference, the proposed method has had a more accurate performance in magnitude prediction with respect to

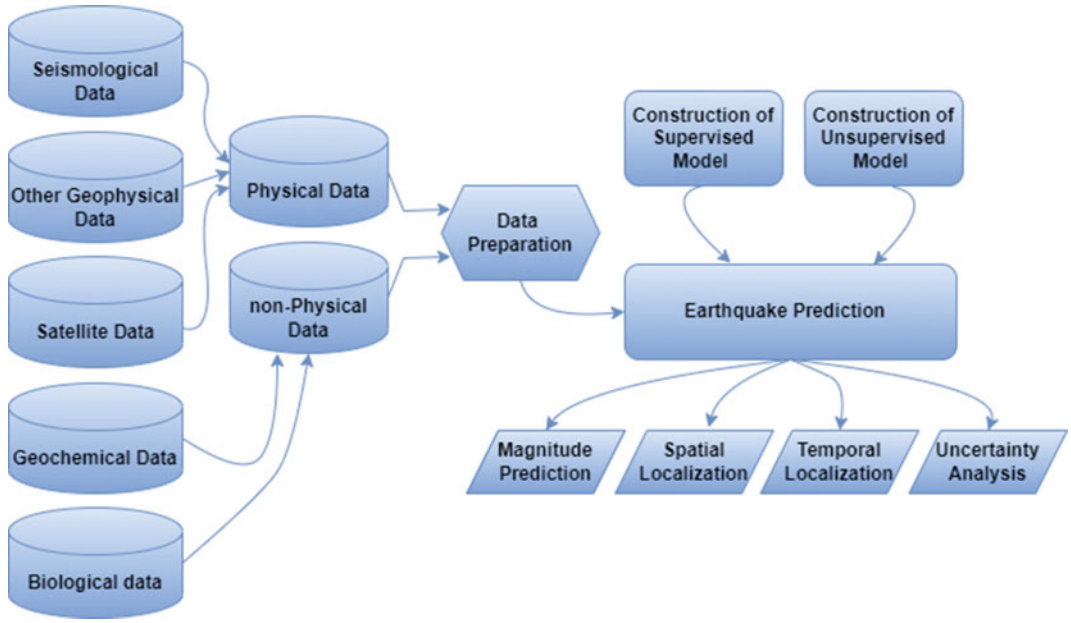


Fig. 1 Demonstration of the general form regarding earthquake prediction problem and the application of machine learning in the approach

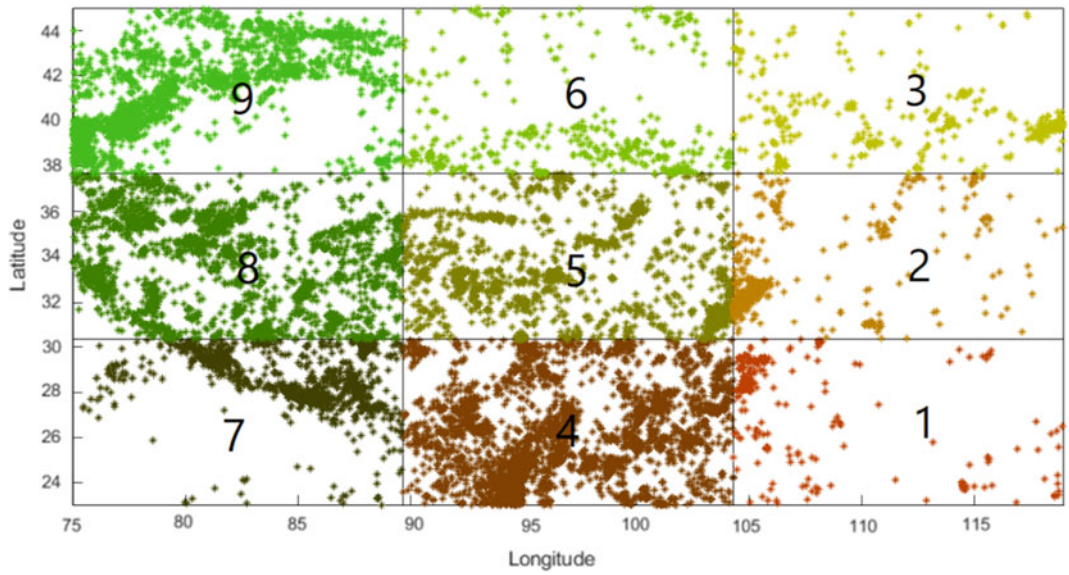


Fig. 2 Epicentral plot of the earthquakes and the division of the study area (Kavianpour et al. 2021)

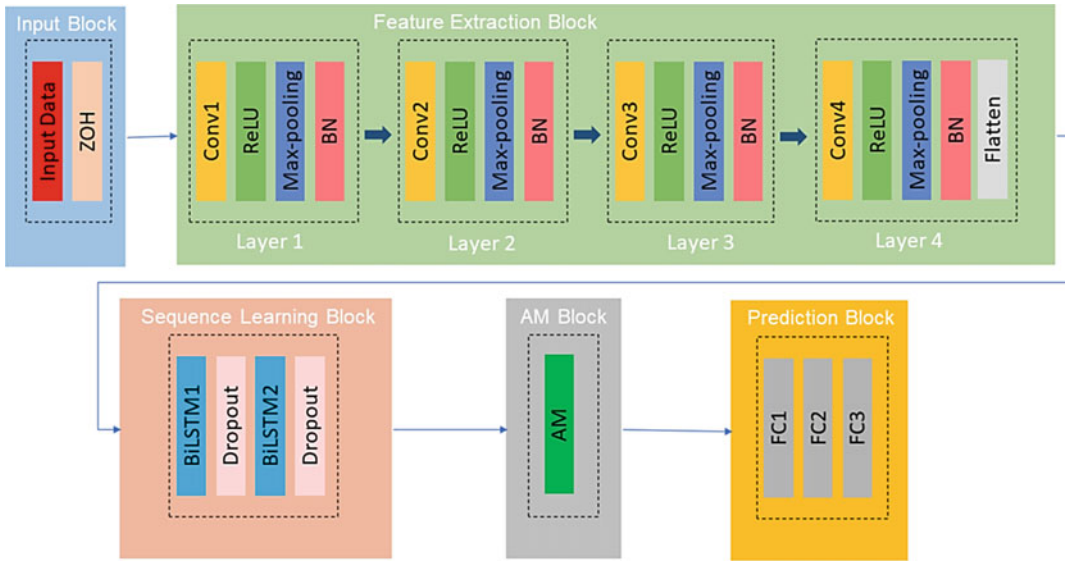


Fig. 3 The architecture of CNN-BiLSTM-AM used by Kavianpour et al. (2021) for long-range-short-term Earthquake Prediction

Table 2 Layer specification of the CNN-BiLSTM-AM model (redrawn after Kavianpour et al. 2021)

Block	Layer	Number/size/stride of kernels or number of neurons
Feature extraction block	Conv1	16 * 5 * 1
	Max-pooling	16/2/1
	Conv2	32 * 3 * 1
	Max-pooling	32/2/16
	Conv3	4 * 3 * 1
	Max-pooling	64/2/1
	Conv4	128 * 3 * 1
	Max-pooling	128/2/1
	Flatten	–
Sequence learning block	BiSTM1	128
	BiLSTM2	64
Attention block	Attention	–
Prediction block	FC1	32
	FC2	10
	FC3	1

Table 3 Some hyper parameters used in CNN-BiLSTM-AM model training

Parameter	Value/range
Optimization algorithm	ADAM
Loss function	Mean square error (MSE)
learning rate	Decreased linearly from 0.001 to 0.0001 by epoch increment
Epoch	150
Batch size	32
Number of iteration	10 times for each subregion

other methods whilst in terms of predicting the number of future events. However, the proposed method still holds the best performance, all DL models are performing acceptably in a range.

2.2 Real-Time Focal Mechanism Determination by Fully Convolutional Network

Kuang et al. (2021) proposed a novel FCN model for focal mechanism determination that is trained by synthetic waveforms generated explicitly for the study area of interest (Ridgecrest region of southern California). The region is first gridded in a 3D manner using 0.1°, 0.1°, and 2 km intervals respectively for latitude, longitude, and depth dimensions. The source is assumed to behave based on the double-couple theory (Aki and Richards 2002) and a 1D model is considered for velocity variations. The synthetic seismograms which are supposed to be recorded at the real location of 16 stations (Fig. 5) are then generated using Thompson-Haskell propagator matrix (Zhu and Rivera 2002) considering all possible variations and combinations of strike, dip, and rake angles in the ranges of 0°–360°, 0°–90°, and − 90° to 90° using 30°, 10°, and 20° intervals respectively for mentioned parameters. Hence, a total number of 787,320 synthetic events has been considered. Each time signal has 128 s length and 1 Hz sampling frequency. Therefore, to be summarized, each event will be corresponded to an input of the shape (16 * 3) * 128 or 48 * 128. Dou to the fact that the real data also may be contaminated with noise and phase picking errors, some additive realistic noise (extracted from real records at each station) and random time shift (< 2 s) have been applied to the synthetics data samples. Furthermore, other mandatory processing operations that are used in conventional real data in seismic networks have been performed on synthetic samples as well as real ones.

As a branch of CNNs, The FCNs, are convolutional neural networks without any included fully connected (dense) layers. The main motivation for proposing and using the FCN networks was to overcome the lack of ability to use input data of different size in CNNs and produce correspondingly-sized outputs with efficient performance. (Shelhamer et al. 2017). In fact, FCNs are capable of transforming the size of intermediate feature maps back to those of the input samples. Figure 6 shows the FMNet, the FCN modified by Kuang et al. (2021) for focal mechanism determination. The left part, the compression or feature extraction part, is fed by 1 * 48 * 128-shape inputs, and using convolutional kernels alongside the max-pooling layers, they are transformed to 128 * 1 * 1 vectors (in the By-product unit). Then, in the right part, or the expansion unit, mentioned vectors are expanded and transformed to 3 * 128 * 1-shape outputs for reconstructing the three Gaussian probability distributions in which the maximum probability of each one, corresponds to the strike, dip, and rake for the determination of the focal mechanism.

Detailed information about the hyperparameters used for training the FMNet model is given in Table 4. Discussion for the procedure of selecting parameters has been provided by the authors in the paper.

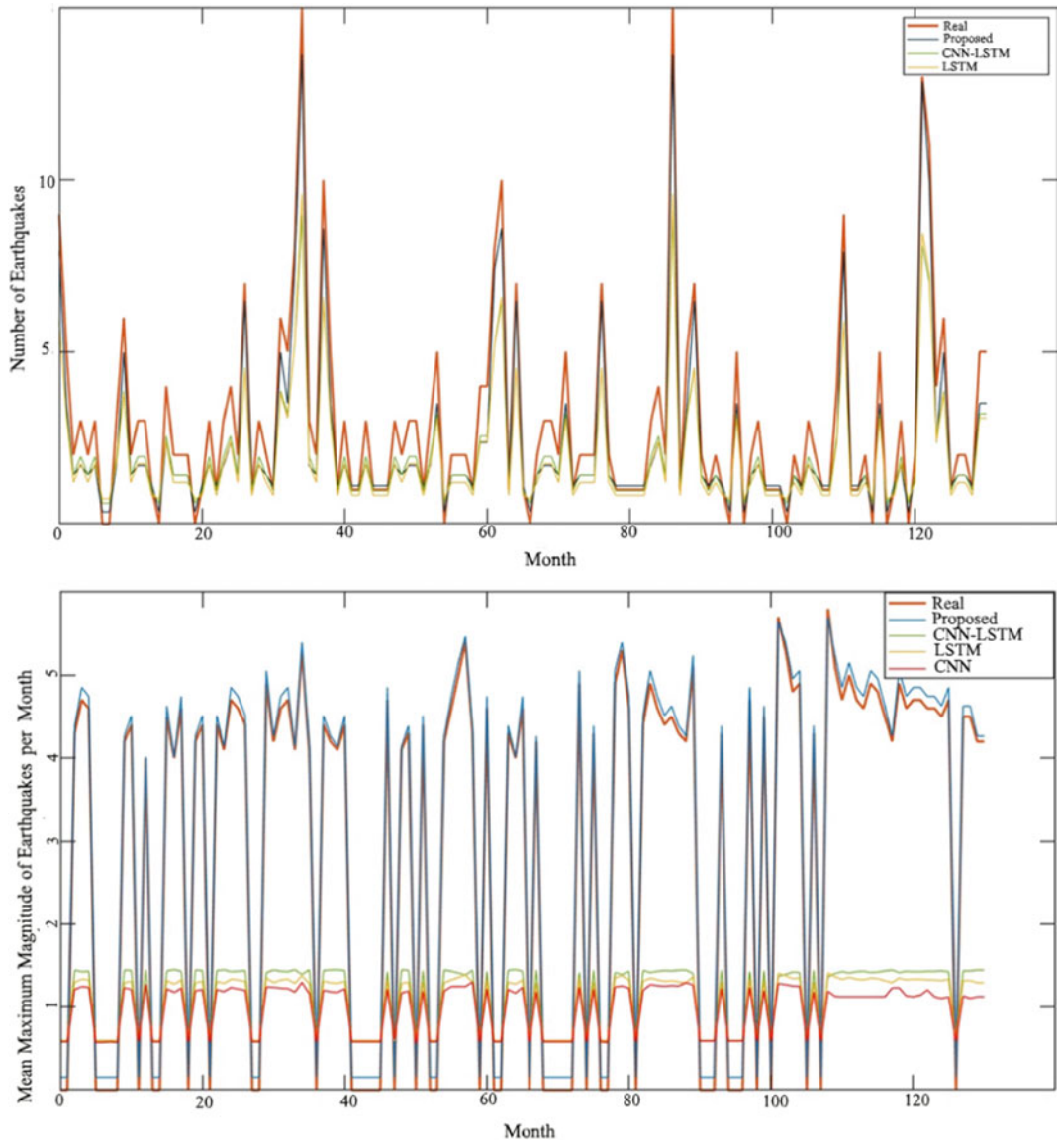


Fig. 4 Real data alongside with the output of the DL models for predicting (top) number of earthquakes for subregion 5 and (bottom) mean maximum magnitude for subregion 1 (Kavianpour et al. 2021)

Four earthquakes ($M_w > 5.4$) have been selected amongst the Ridgecrest earthquake sequence which was proceeded by an Mw 6.4 foreshock and followed by an Mw 7.1 mainshock.

The interpreted results of generalizing the trained model to the test data are illustrated in Fig. 7 by red-colored plots and a strike-slip faulting mechanism with very steeply dipping fault planes is revealed. Compared to the focal mechanisms determined from the generalized cut-and-paste method (Zhu and Helmberger 1996) for the northernmost event, and moment tensor catalog analysis for the three earthquakes in the southern region, the results of FMNet are consistent. Compared to other methods and studies, the proposed method needs less than a second to solve the focal mechanism problem and thereby supports the applications of the

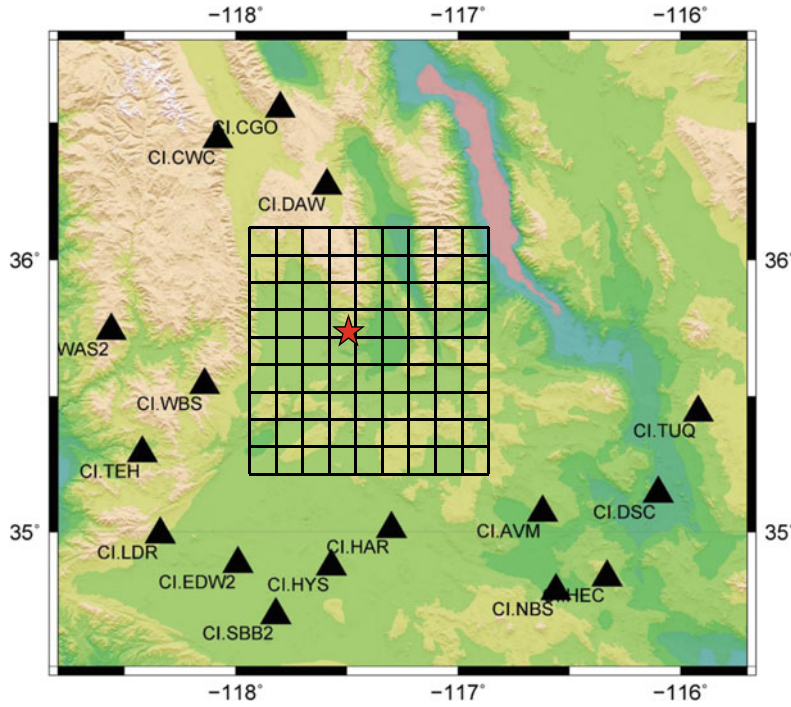


Fig. 5 The location of area (gridded region) used for synthetic event generation and 16 seismic stations (black triangles). The red star denotes the mainshock of the July, 2019, Ridgecrest earthquake sequence (Kuang et al. 2021)

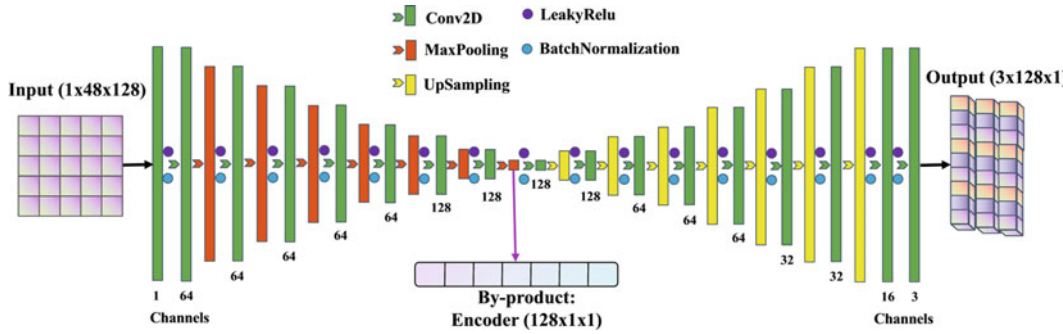


Fig. 6 The FMNet architecture (Kuang et al. 2021)

Table 4 Some hyper parameters used in FMNet training

Parameter	Value/range
Optimization algorithm	ADAM
Loss function	Mean square error (MSE)
learning rate	0.001
Epoch	50
Batch size	16

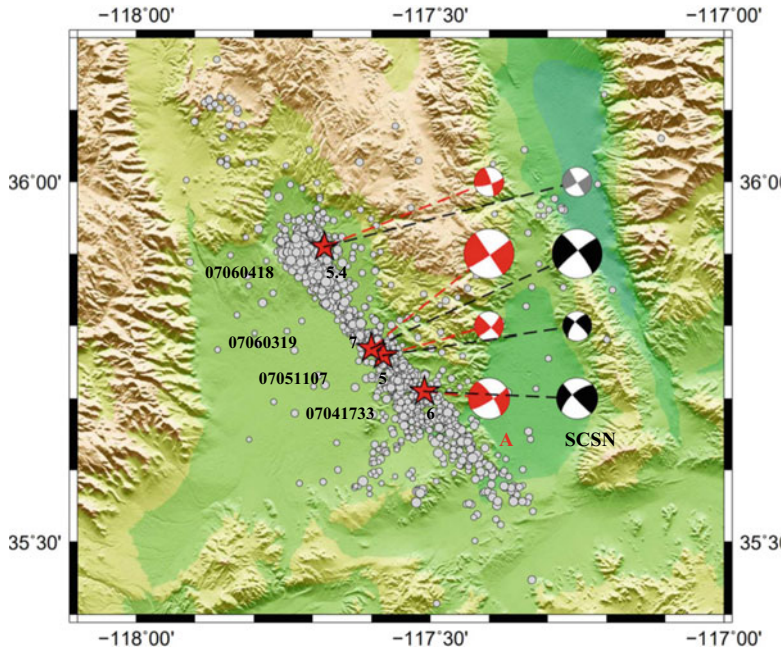


Fig. 7 Interpretation resulted from the output of generalizing the trained FMNet model to the test data. The output of the FMNet and other conventional methods (specified in the text) are plotted by red and black/gray colors respectively (Kuang et al. 2021)

proposed method for different aspects in the field of seismic hazards studies.

2.3 A Functional Very Deep Convolutional Neural Network Model for Fast and Precise Earthquake Phase Picking

In the valuable research of Wang et al. (2019) a functional DL model is developed based on a very deep CNN model. As the main difference with more simple sequential models, the functional structures are more flexible and usually more powerful and this is mainly due to the possibility of using arbitrary connections between layers or on the other hand, the possibility of having multiple inputs and outputs. In this section, the main focus will be on reviewing the network architecture.

In the previous chapter, a few sequential models were defined and used for solving some problems. Here, an example of a functional model is shown to acquaint the dear readers. The inputs to the model are supposed to be three-channel 128 * 128 images. After generating the model using the code provided below, the resulting model is illustrated in Fig. 8:

```
#Loading required modules and libraries.
from keras.utils.vis_utils import plot_model
from keras.models import Model
from keras.layers import Dense,Input, Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPooling2D
from keras.layers.merge import concatenate
#The Keras Functional API allows using input from previously defined layers.
#In doing so, after defining. For every new layer, the input to the layer is
#determind just by naming the input(128*128 RGB images)in parentheses.
input = Input(shape=(128, 128, 3))
```

```
con1 = Conv2D(filters=32, kernel_size=3, activation='relu')(input)
pool1 = MaxPooling2D(pool_size=(2, 2))(con1)
con2 = Conv2D(filters=16, kernel_size=5, activation='relu')(pool1)
pool2 = MaxPooling2D(pool_size=(2, 2))(con2)
flat1 = Flatten()(pool2)
dns1 = Dense(50)(flat1)
flat2 = Flatten()(pool1)
dns2 = Dense(50)(flat2)
#Merging final layers for the creation of the output
concat = concatenate([dns1, dns2])
output = Dense(1, activation='sigmoid')(concat)
#At the last step of the mdel definition, the initial input and final output
#must be specefied
func_model = Model(inputs=input, outputs=output)
#Plotting the model's graph with in ond out shapes
plot_model(func_model, show_shapes=True )
```

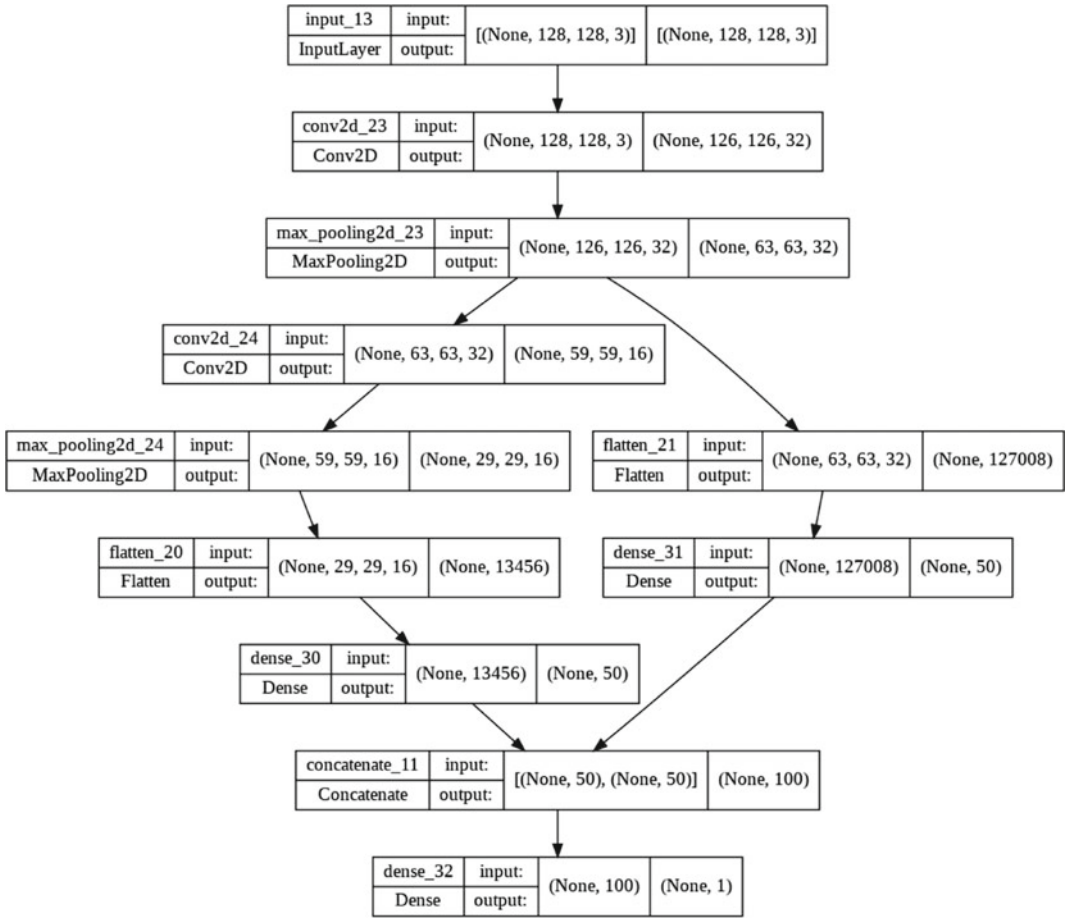


Fig. 8 The graph of a functional model. The "None" parameter indicates unspecified batch size

Detailed and comprehensive information about Picknet, the proposed model by Wang et al. (2019), is illustrated in Fig. 9, where, all functional connections are also plotted for the entire architecture. Five sequential blocks of convolutional layers are used in the Picknet each of which is connected to side layers and sequential connections. Mentioned side layers or "side-outputs" are rich side-output residuals networks (Simonyan and Zisserman 2014) combined with residual calculator operators and fed by real data samples. Also, as shown on the

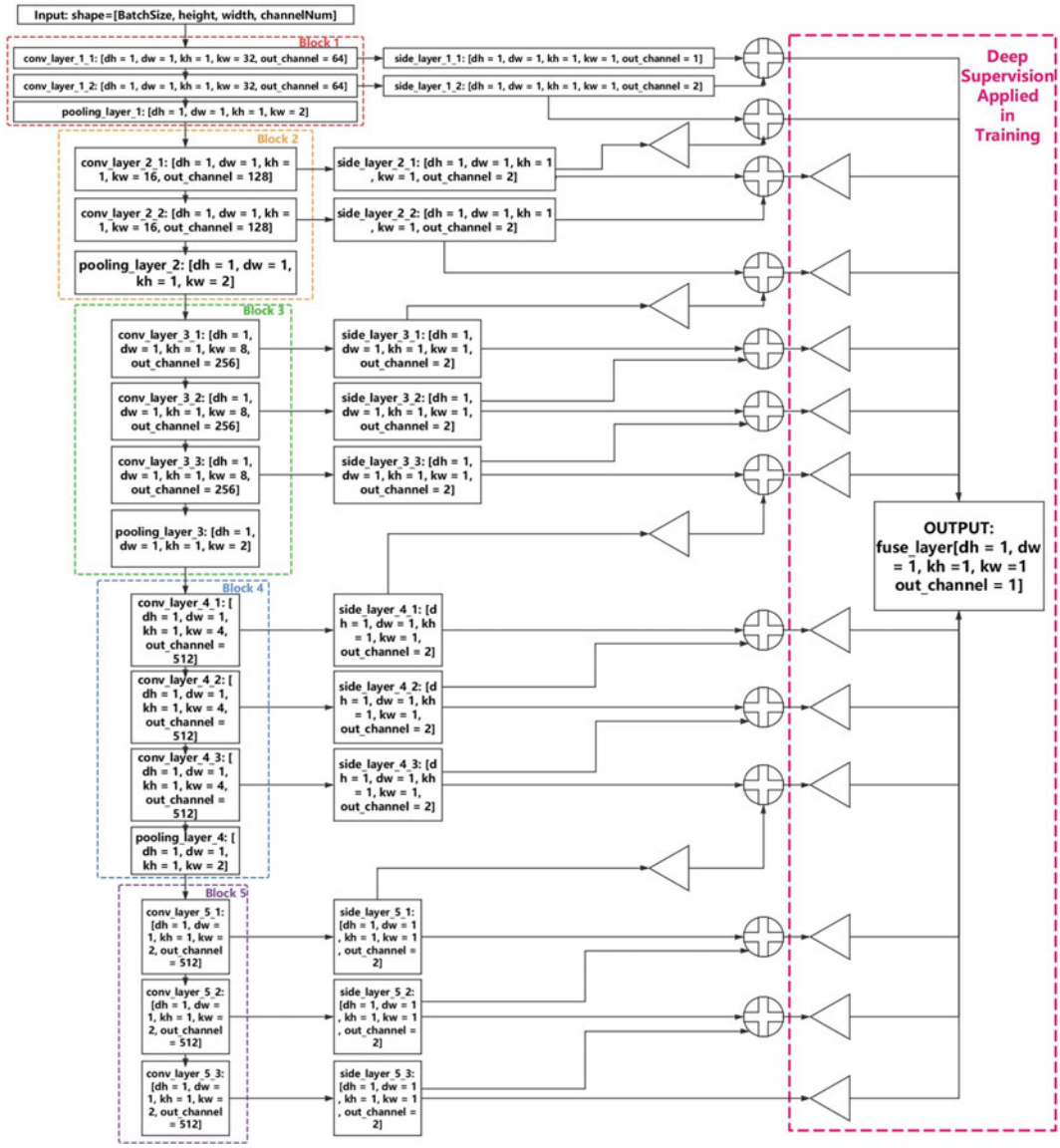


Fig. 9 The architecture of the PickNet. Here, dh, dw, kh and kw respectively denote stride on height, stride on width, kernel height and kernel width values. out_channel stands for the number of output filters in the convolution layer. Triangles and circled plus show the up-samplings by deconvolution and the residual units respectively. In the deep supervision block, all tensors at the end of every branch are also supervised with the targets using the same loss function as final output before being fused (Wang et al. 2019)

rightmost side of the figure, a multiscale combination strategy using convolutional layers is used to generate the final output.

The training inputs are somehow different from similar studies, here, vertical, radial and transversal components have been used different signal lengths. The vertical component information is used for P wave picking and the others are used separately for S wave picking. There so, two different models have been trained.

Data preparation includes centering the waveforms on their theoretical arrival times using TauP code (Crotwell et al. 1999) and also normalization. Manually phase pickings are used as real data targets for training set.

Picknet has been trained with about 460,000 P and 280,000 S wave samples that all have been manually picked with high accuracy from three-component seismograms recorded at 782 stations of the dense High sensitivity Seismic Network (Hi-net) in Japan. Also, a set of 300 local earthquakes in Japan as well as events recorded by the China Earthquake Administration, the International Seismological Centre, and the Southern California Earthquake Data Center are used for performance evaluation. Two samples of the mentioned datasets are illustrated in Fig. 10. Details about parameters used in the training procedure are shown in Table 5. A total number of 97,998 P and 92,229 S waves arrival times has been obtained from analyzing mentioned 300 events. In contrast, Japan Meteorological Agency (JMA) has only reported 13,765 P wave and 8643 S wave arrival times.

The authors have also conducted a tomographic inversion for 3D, V_p and V_s determination in the Japan subduction zone based on the results of the phases picked by PickNet using Tomog3d package (Zhao et al. 2012).

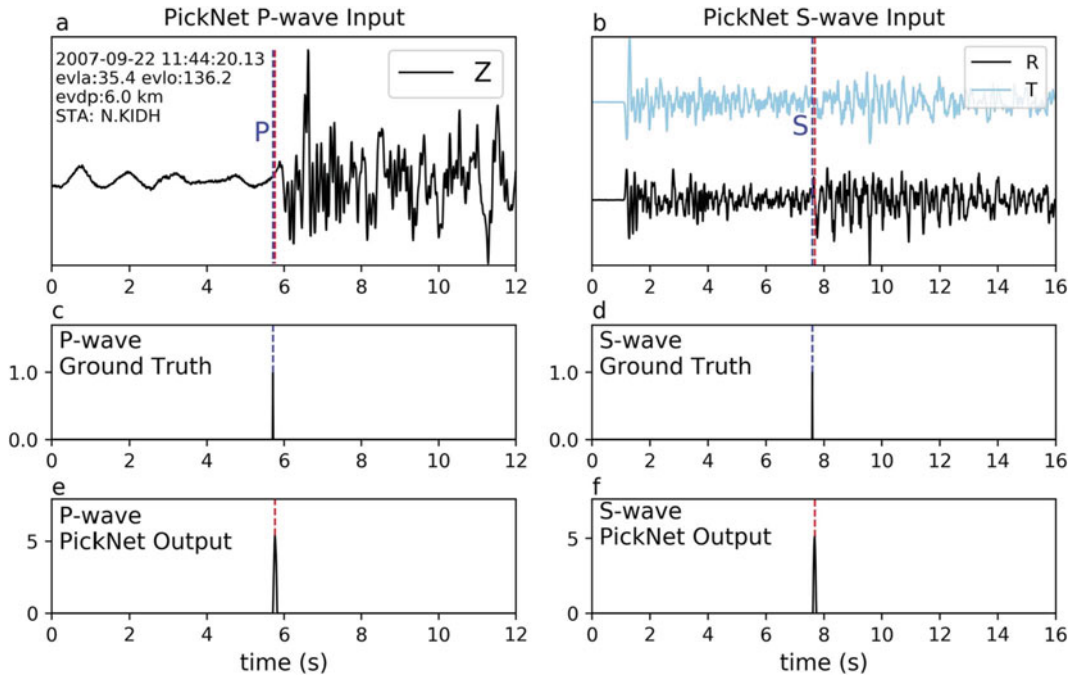


Fig. 10 Results of generalizing the trained PickNet models for P (a, c, e) and S (b, d, f) phase determination. The waveforms are recorded at station N.KIDH in Japan

Table 5 Hyper parameters used in the PickNet training procedure

Parameter	Value/range
Optimization algorithm	ADAM
Loss function	Cross-entropy
Epoch	Not specified, instead, steps are stated as about 350,000 and 800,000 respectively for P and S wave models
Batch size	400 for P and 50 for S waves

2.4 Magnitude Calculation Directly from Raw Waveforms Using MagNet

A regressor deep model named MagNet composing CNN and BiLSTM without sensitivity to data normalization is proposed by Mousavi and Beroza (2020). The main considerations in the development of the method include reliability and reducing the time of the calculations. Among the achievements of the research, simple data preparation could be considered as a great novelty since the raw seismogram waveforms are fed to the network just after simple band-pass filtering and even, there is no need for removing the instrumental effect from the records.

The architecture of the MagNet consists of two CNN followed by a BiLSTM layer (Fig. 11). Dropout and max pooling layers are also used after each convolutional layer. It must be noted that convolutional blocks are not equipped with any activation function and this is due to their usage in this model, since, they are supposed to extract features and reduce the dimensionality of the inputs. The learning process is mainly accomplished in the BiLSTM layer and following one neuron fully connected layer.

As the seismogram response has not been removed from the waveforms, a mechanism for this deficiency must be considered. As stated by the authors, using three-component waveforms that are long enough to promise the coda wave information is fully recorded could be a solution for overcoming the problem. In doing so, any used waveform is here 30 s long.

The dataset used in the research is composed of about 300,000 waveforms whose epicentral distances are less than 1°. The waveforms are trimmed from 1 s before P arrival until the end of the S coda and their length is limited to 30 s. The

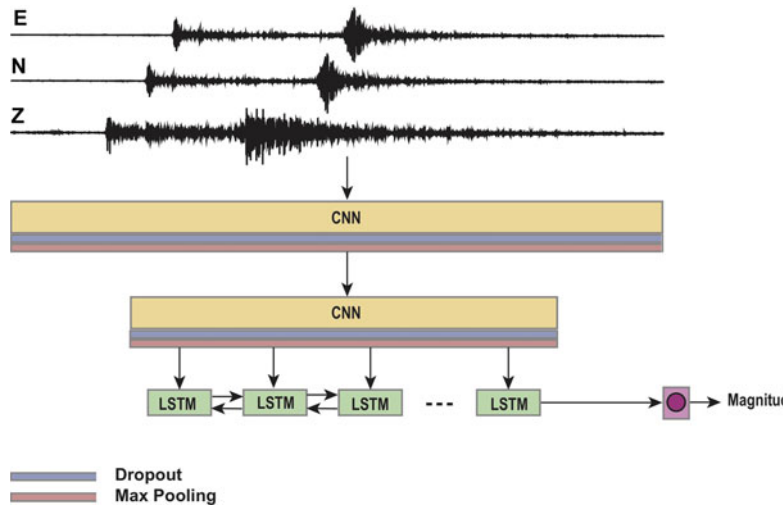


Fig. 11 The MagNet Architecture. First and second convolutional layers are of sizes 64 and 32 and kernels of size 3 respectively. There are 100 units in the BiLSTM layer (Mousavi and Beroza 2020)

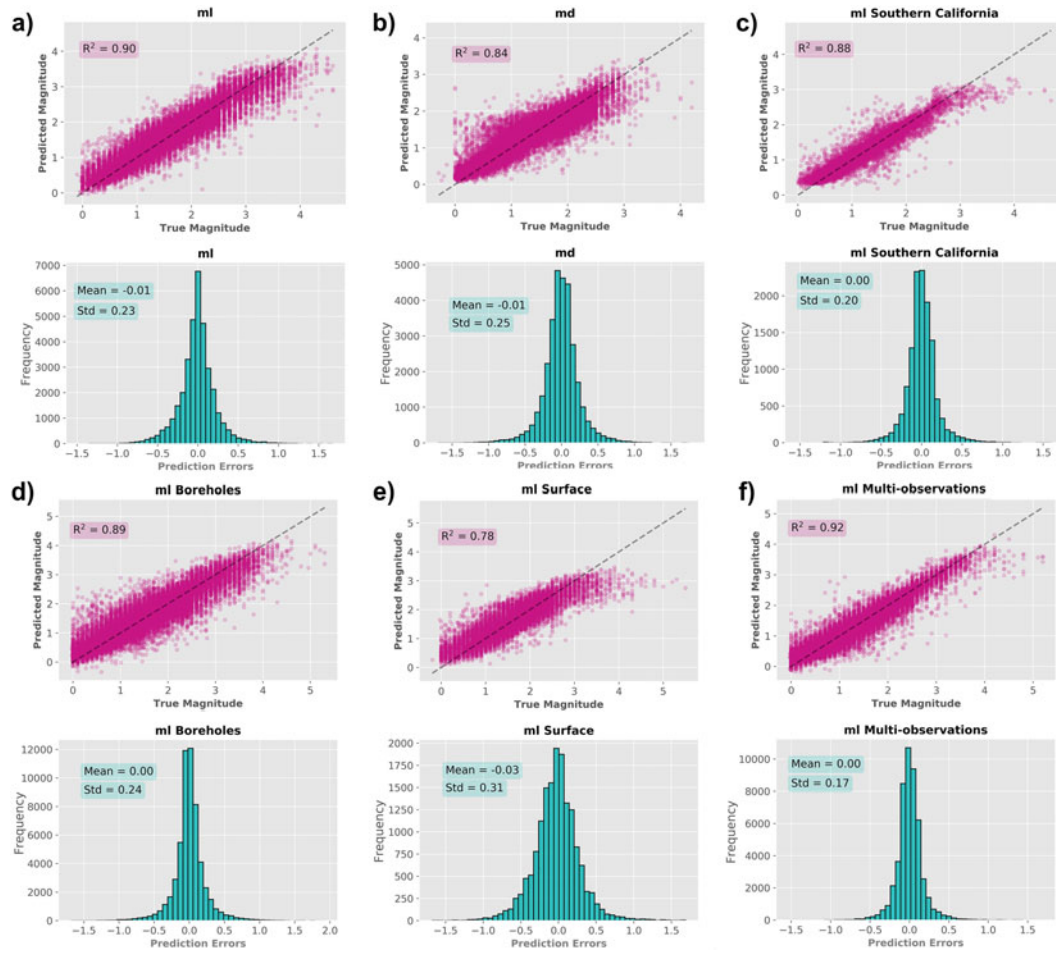


Fig. 12 Magnitude calculated by MagNet results on test datasets for **a** local magnitude (m_L), **b** duration magnitude (m_d), **c** events that occurred in the Southern California, **d** event recorded with borehole stations, **e** surface stations, and **f** in a case where only stations with more than 1000 observations are used (Mousavi and Beroza 2020)

dataset is divided into smaller subsets for precisely investigating the role of effective parameters such as magnitude, SNR, etc. each set is then portioned to training (70%), validating (10%), and testing (20%) subsets. Details about training hyperparameters that are mentioned in the text, include optimizer (Adam) and criteria for stopping the training procedure, which is the case when validation loss does not decrease for five consecutive epochs.

The calculated magnitudes versus true values regression plots are illustrated in Fig. 12. The authors conclude that the method can calculate magnitude, based on single-station waveforms with an average error close to zero and a standard deviation of about 0.2.

Also, the authors provided an analysis of affecting parameters that reduce the efficiency of the method and found that the signal-to-noise ratio has the most significant impact among all investigated parameters. Variation of prediction error versus SNR and also the depth of earthquakes are illustrated in Fig. 13.

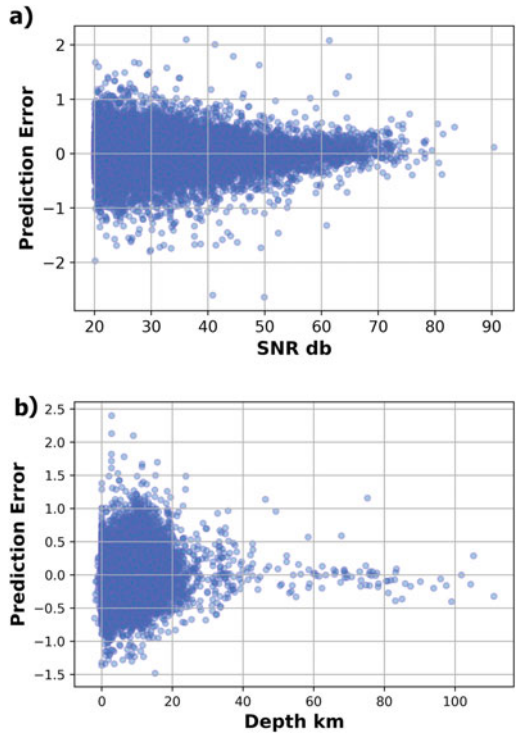


Fig. 13 The MagNet prediction errors as a function of **a** SNR and **b** depth of events (Mousavi and Beroza 2020)

3 Detecting and Locating Induced Seismicity Using ConvNetQuake Model

Minor earthquakes and tremors caused by the human-activity are called Induced seismicity, which changes the stress and strain regime of the crust. Induced quakes are usually of low-sized magnitudes and are spatiotemporally localized. In Fig. 14, the concepts associated with induced earthquakes, commonly analyzed in related studies, are displayed.

According to the increase of various sources of induced seismicity, monitoring and analyzing resulted earthquakes are among the considerations of many researchers. A method aiming to be fast, scalable, and provide outputs based on single station waveforms is presented by Perol et al. (2018) in the mentioned field. They presented a CNN model, named ConvNetQuake, for detection and location of earthquakes and trained and evaluated the model using earthquakes mainly related to a stimulation operation performed in Oklahoma, USA, and occurred between 15 February 2014 and 16 November 2016.

The ConvNetQuake model consists of eight convolutional layers followed by one flatten and one fully connected layers (Fig. 15). The data provided for the research corresponds to Guthrie, Oklahoma, which is an active seismic region due to the waste water injection projects. A total number of 2918 earthquakes extracted from three-dimensional 100 Hz ground velocity streams recorded by two seismic stations were used. Based on behind logic of the method, also, a set of 831,111 noise data is used to fill the gaps between mentioned cataloged events. Due to the imbalanced ratio of the portion of events and noise in the resulting dataset, an augmentation procedure has been used for the events-related elements.

Each input is 10 s long and contains three components of the waveform. The study area has been partitioned into six regions (Fig. 16) and the region number is attached to corresponding events. The model is then trained using 90% of the prepared dataset and the remaining 10% is used for evaluation.

The earthquake detection accuracy is studied based on true noise and event-labeled outputs. All of the events (100%) were detected by the model while, 94.8% of the noise data were detected correctly and the rest were misclassified. The locating accuracy is investigated through studying the region classes outputted by the model, which, resulted that the model has correctly predicted the location of 74.5% of the total events within the test data.

Alongside other discussions provided by the authors in the research, using synthetic seismograms, a comparison between the proposed method and the template matching algorithm is performed mainly to investigate the efficiency of methods, in locating nonrepeating events. As illustrated in Fig. 17, the accuracy of the ConvNetQuake model increases gradually by SNR increment, while, for the template matching, no meaningful relationship could be observed.

As stated by the authors, because, after training the deep ConvNetQuake model, the

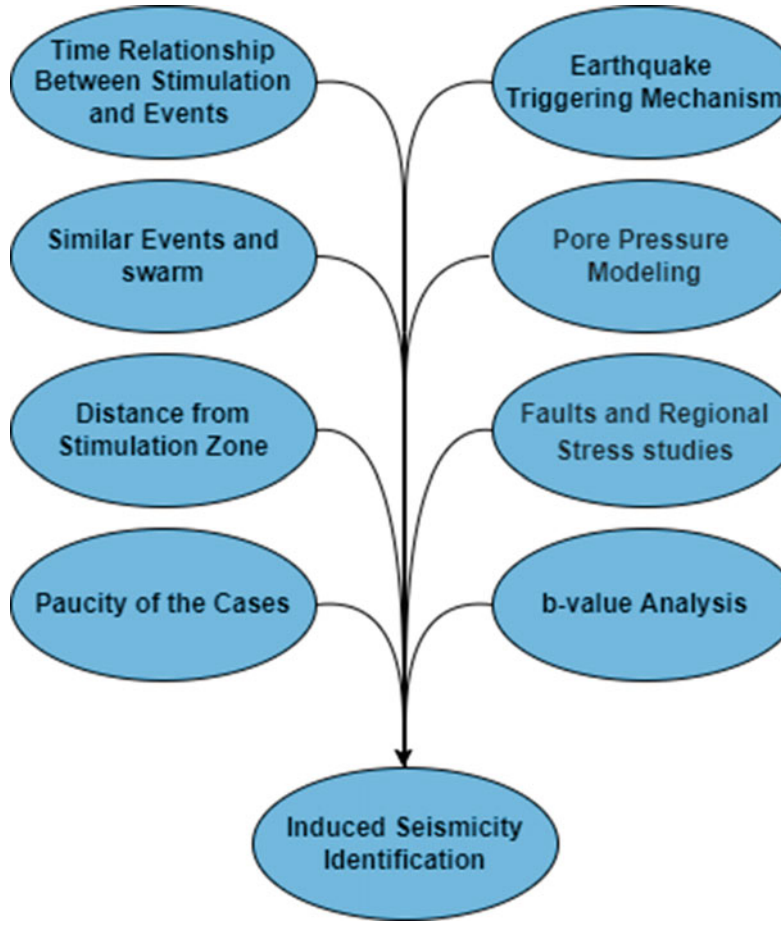


Fig. 14 The concepts associated with induced seismicity

generalization procedure could be achieved very fast, the proposed method could be considered as a powerful method for earthquake detection and location problems which could be beneficial for being sued in earthquake early warning systems. Furthermore, it could effectively apply to many other issues such as monitoring geothermal resources, reservoir studies, volcanology, etc.

3.1 Classification Based on Limited Training Samples Using CapsNet: Aplication to Microseismic Record Classification

A method for classifying microseismic events is presented based on the motivation to develop an automated process less dependent on a large training dataset. The model used in the precious research of Peng et al. (2020) is based on the CapsNet architecture that is a capsule neural network (Bonheur et al. 2019) and as shown in Fig. 18, after the convolutional unit including a max-pooling layer, "primary capsule" and "digit capsule" layers are applied. The mentioned layers are used to empower the architecture ability in spatial feature extraction, since, the capsule units are capable of providing vectorized information instead of outputting a scalar, like how a neuron does. This ability is beneficial, especially when the orientation of elements in inputs changes. That is why usually, the necessary computational resources reduce the training procedure of CapsNet.

Five training sets, of sizes, 400, 800, 1,200, 1,600, and 2000 records, from microseismic data acquired at a Copper and Zinc mine, are selected for training the model with a 20% split, as the

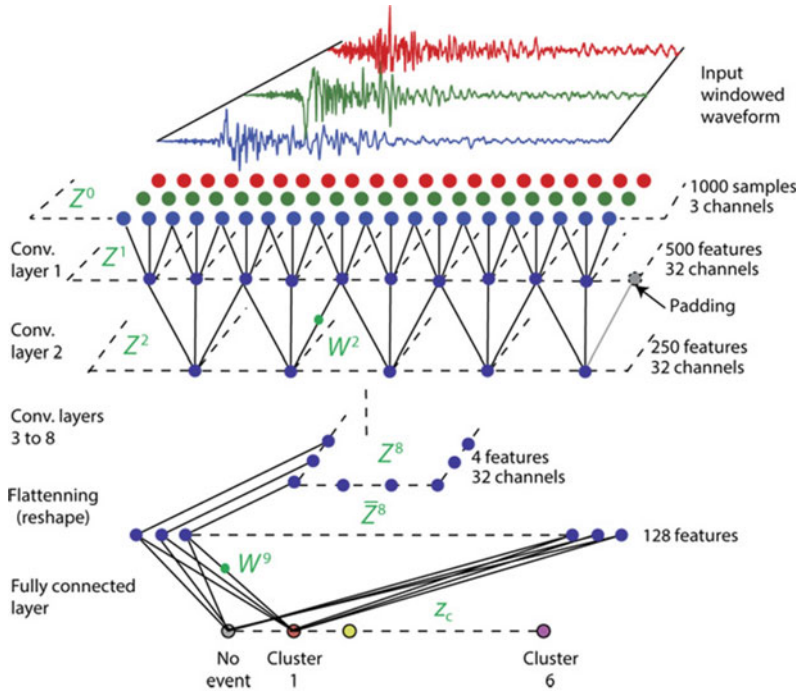


Fig. 15 The architecture of ConvNetQuake model consists of eight convolutional layers, one flatten and one fully connected layer. The output of the model indicates whether the input corresponds to noise or an event related to any of clustered (Fig. 16) regions (Perol et al. 2018)

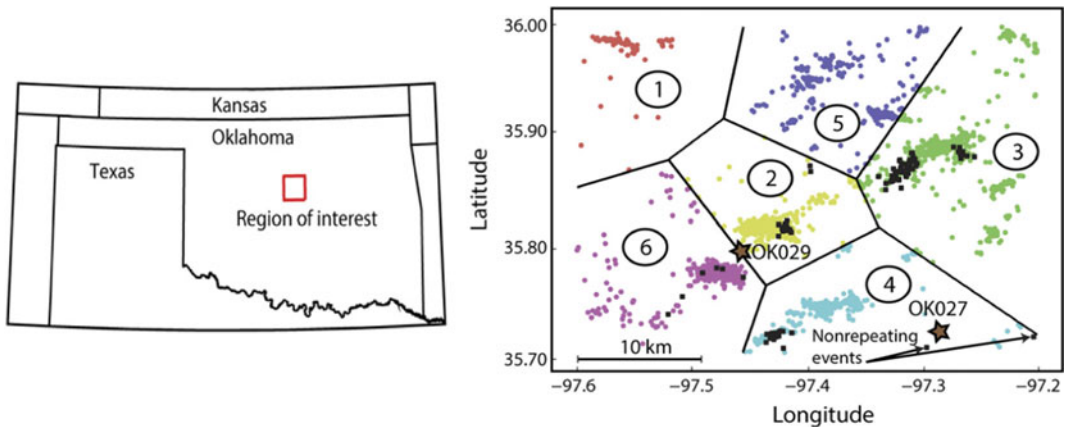


Fig. 16 (Right) Earthquakes (circles), seismic stations (stars) and the partitioned regions used in the study (left image, Oklhoma, USA). The black squares indicate the non-repeating events used for template matching method (Perol et al. 2018)

validation subset. Five different model training scenarios were utilized, which are detailed in Table 6. Applying different training configurations is mainly due to quantitatively investigating the method's performance in the cases where the training data is relatively small.

As input data, a matrix constructed using 21 commonly applied features, extracted from a

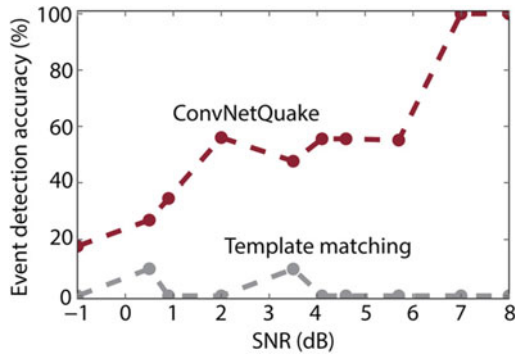


Fig. 17 Detection accuracy for ConvNetQuake and template matching methods (Perol et al. 2018)

major microseismic event, is used. Since each record (10,000 samples) is divided into 33 frames, for each the related input is of size 21 * 33. Inputs are set to belong to one of the four classes labeled as microseismic events, blasts, ore extraction, and noise. The mentioned procedure is schematically depicted in Fig. 19.

After training the CapsNet model (as well as a CNN model for performance evaluation), it is evaluated using separate data that contains 3200 microseismic events. In Fig. 20 the accuracies of both models' classifications for the five mentioned training processes are shown. The authors provide detailed and comprehensive discussions, including the implementation of another test dataset. Finally, based on the revealed results, it is concluded that the proposed method is superior to compared methods even with smaller training databases.

3.2 Deep Convolutional Neural Network for Fast Prediction of Earthquake Intensity from Raw Accelerograms

Jozinović et al. (2021) proposed a novel method for earthquake intensity prediction based on the purpose that the method be fast enough and reliable to be used in real-time earthquake data stream analysis. They used the three-component waveforms corresponding to 700 earthquakes that occurred between 1 January 2016 to 29 November 2016 and 1037 noise-related samples

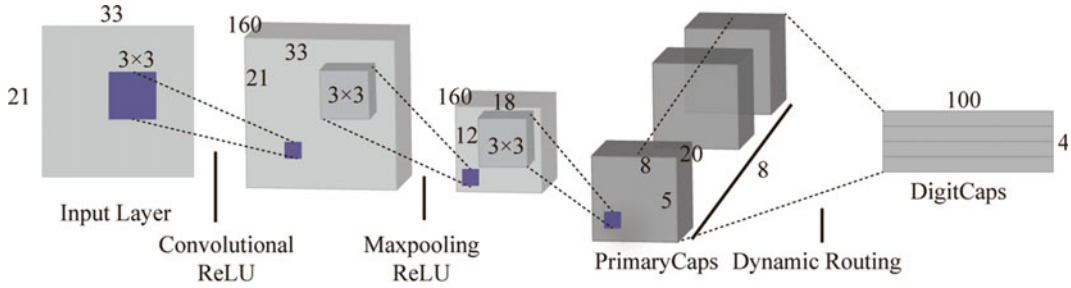


Fig. 18 Detection accuracy for ConvNetQuake and template matching (Peng et al. 2020)

Table 6 Different model configuration used for microseismic classification (redrawn after Peng et al. 2020)

Process	Amount of data used for training	Amount of data used for validation	Amount of data used for the test
Training process 1	320	80	3,200
Training process 2	640	160	3,200
Training process 3	960	240	3,200
Training process 4	1280	320	3,200
Training process 5	1600	400	3,200

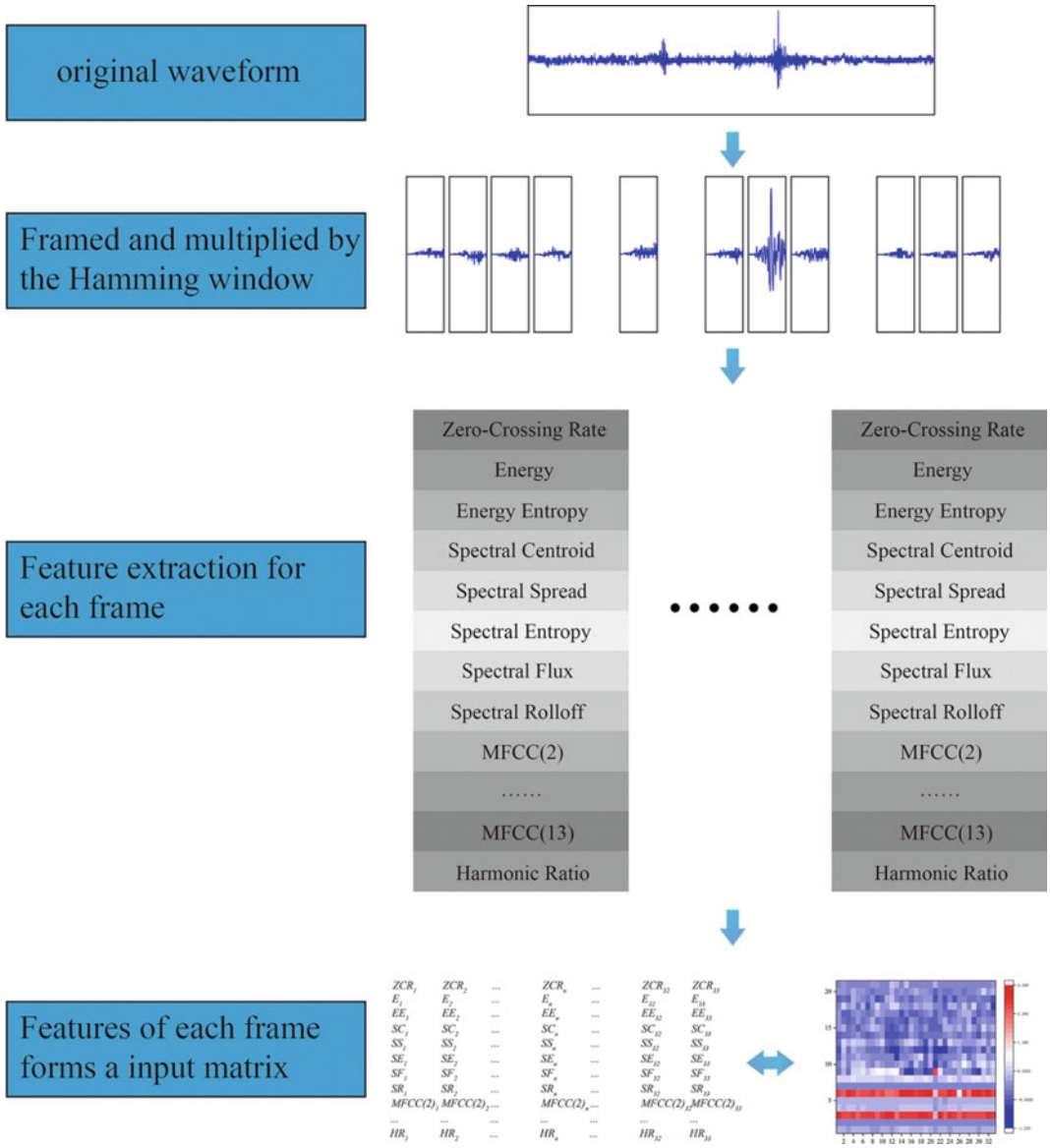


Fig. 19 Data preparation from raw microseismic events (Peng et al. 2020)

recorded from 30 August to 30 September 2016. All were recorded in 39 stations of the IV network (Italy). Mentioned data were resampled to 100 Hz and normalized. The proposed method declared to use raw accelerometers as inputs, there so, in case, velocity records were provided by the seismic network, conversion operation to accelerometer has been used. Earthquake epicenters, network stations and the area related to the study are shown in Fig. 21.

For each earthquake, 10 s frames (starting at earthquake origin time) of all three-components waveforms recorded at 39 stations are fed to the network simultaneously as inputs and the corresponding ground motion parameters (PGA, PGV, spectral acceleration at 0.3, 1 and 3 s periods) are set as targets. For the noise waveforms, the maximum amplitudes recorded at the station inside the window is used as input. Since the normalized versions of waveforms are used as

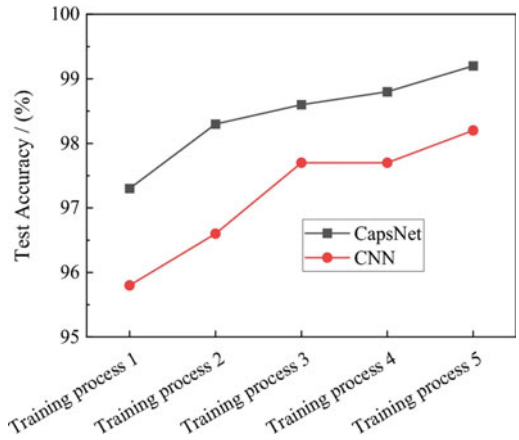


Fig. 20 Data preparation from raw microseismic events (Peng et al. 2020)

inputs and the maximum amplitude recorded at each station physically is related to strong motion parameters, the normalization factor of each waveform is also fed to the last flatten layer after being concatenated with the output of the CNN sequence (Fig. 22).

The CNN model is trained using 80% of prepared data as train and the rest, as validation sets and the resulting generalized outputs are studied through different methods and using multiple experiments. For instance, the frame length is investigated using three window sizes: 7, 10 and 15 s windows. and results are shown in Fig. 23.

Based on the results of miscellaneous experiments provided by the authors, it is concluded that the proposed method is capable of accurately predicting earthquake intensities at stations, even for the points that are far from the epicenter and for which have not yet recorded the maximum ground motion. Also, it is found that the method could provide estimates of ground motions within 15–20 s after earthquake origin time.

4 Applications of Deep Learning in Volcanology

Here, like the study pattern performed in Sect. 1, a target-based schema is used to review DL models' applications in volcanic studies. As the significant concepts, eruption prediction, activity classification, and providing estimation about magma-related approaches, such as chamber determination and magma dynamics, etc. were considered initially, but based on the fact that many researches, simultaneously cover all or

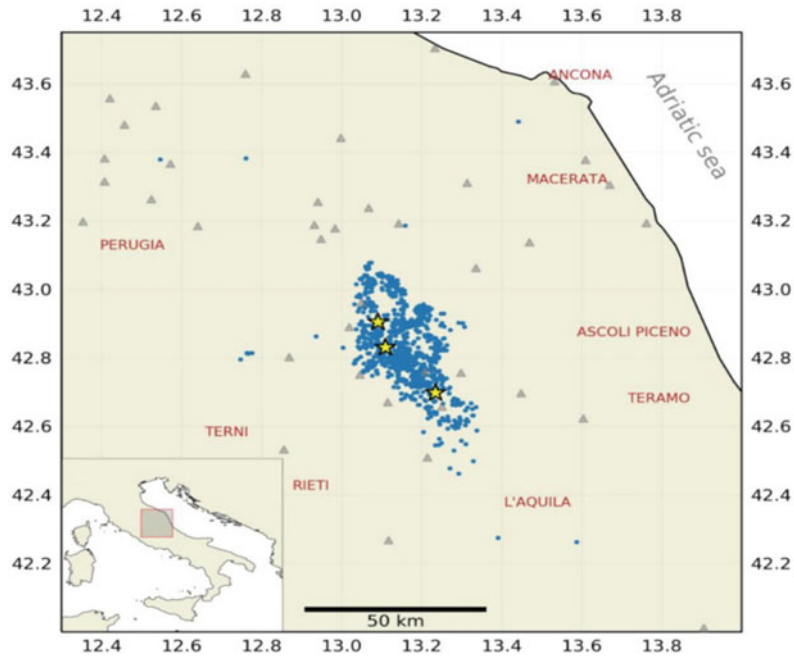


Fig. 21 Earthquake epicenters, network stations and the area related to the study (Jozinović et al. 2021)

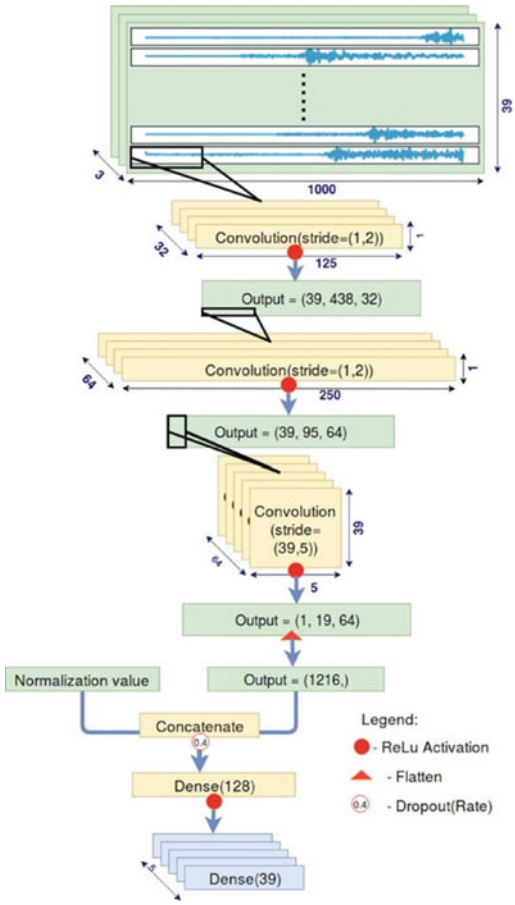


Fig. 22 The architecture of the model used for rapid strong motion prediction (Jozinović et al. 2021)

most of the mentioned concepts and also considering that, to best of our knowledge, there is no published research specifically for the third major category, all reviewed researches are considered to belong to the category of volcanology (Table 7). Again, a brief summary has been provided for all research and the selected rows (starred items); additionally, a detailed review has been accomplished.

4.1 Volcano Deformation Identification Using Convolutional Neural Network Trained by InSAR Synthetic Database

Since there is usually concerns about the data imbalances and also presence of atmospheric noise in the instances of InSAR data, when are applid in ML models, Anantrasirichai et al. (2019) presented a method to detecting volcano deformations utilizing a synthetic dataset.

Three classes of deformations were considered to be identified by the method: (1) seismologically related deformations including earthquakes, dykes, sills and point pressure changes at magma chambers, (2) stratified atmospheric artifacts and (3) turbulent atmospheric artifact. Corresponded synthetic samples are generated using methods described in the paper in detail. For each class of deformation, 10,000 were generated each of which has a 500 by 500 pixels dimension. In Fig. 5.24, 15 samples of the generated synthetic InSAR images are depicted. In addition to the generated synthetic database, a real InSAR dataset acquired by the

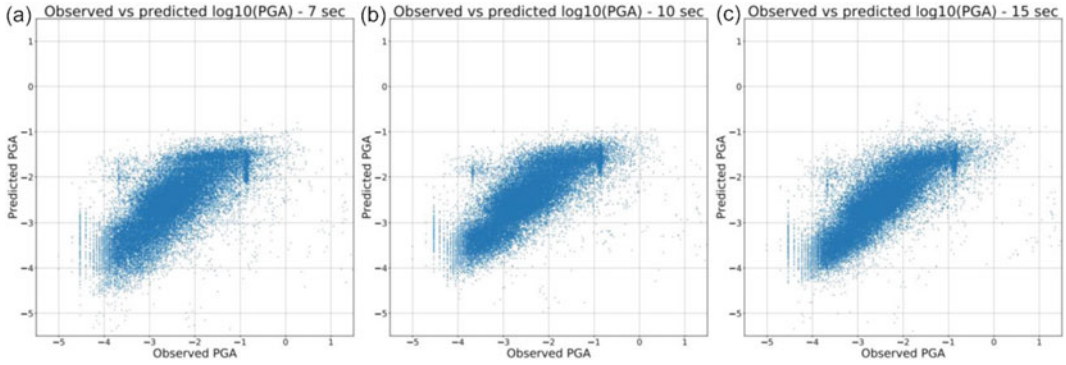


Fig. 23 Regression plot of predicted versus observed values resulted by using different frame size, **a** 7 s length, **b** 10 s length and **c** 15 s length. The Log10 of the values are used (Jozinović et al. 2021)

Table 7 Brief review of some examples using DL models in volcanology

Article no.	1	The research is presented based on the motivation of performing uncertainty analysis, alongside using the DL method for volcano studies. Authors have developed a deep Bayesian neural network that is utilized for seismic waveform identification and classification and also corresponding uncertainty analysis has been provided. The earthquake waveforms (inputs) during eruptive periods, are classified into three labels, namely low, mixed high-frequency earthquakes. The first two labels considered corresponding to mechanisms such as volumetric sources, magma fracture, stick–slip along the margins of volcanic conduits, etc. whilst, high-frequency events are considered to probably be related to magma "propagation through strongly attenuating volcanic material". The method has been performed on datasets related to Mount St. Helens (USA) and Bezymianny, volcanos (Russia) simultaneously. An accuracy rate of 92.08% has been resulted (utilizing 20% of the dataset as test data) by the proposed method and it is stated that the proposed model is capable of indicating different eruptive periods and related estimated uncertainty corresponds to changes in the state of unrest at the volcanoes
Article title	Volcano-seismic transfer learning and uncertainty quantification with bayesian neural networks (Bueno et al. 2020)	
Used model	Deep bayesian neural network	
Article no.	2	A feasibility study along with providing an on-board prototype (based on the Raspberry PI controller) model is presented and evaluated for eruption identification. Two CNN (one has smaller architecture) have been developed and studied for on-board requirements. The 512 by 52 images prepared using Landsat-7 and Sentinel-2 data related to five eruptions are used for training the model: Ulawun (26 June 2019), Ubinas (24 June 2019), Raikoke (22 June 2019), Piton de la Fournaise (11 June 2019) and Great Sitkin (1 June 2019). Two labels were used to train the model, namely "eruption" and "no eruption," corresponding to 334 and 818 instances. The validation dataset also contains 75 "eruption" and 94 "no eruption" samples. The performances have been evaluated using drone captures (from printed satellite images) instances, and 85% and 83% of accuracy have been reported for respectively big and small architecture CNN models
Article title	On-board volcanic eruption detection through CNNs and satellite multispectral imagery (del Rosso et al. 2021)	
Used model	CNN	

(continued)

Table 7 (continued)

Article no.	3	A DCGAN model developed and called ESeismic-GAN, is utilized aiming to generate the magnitude frequency response of events such as long-period and volcano-tectonic events. Seismic signals are transformed to furrier domain and magnitude responses are set as Gan inputs and the phase information alongside the Gan's output are used for inverse transformation. The training data consists of 1044 long periods and 101 Cotopaxi (Ecuador) volcano-related samples. The developed model is evaluated during training and after signal reconstruction using Frechet distance and Frechet inception distance, respectively, demonstrating that ESeismic-GAN can be used in volcanic studies such as characterizing long-period volcano-tectonic events effectively
Article title	ESeismic-GAN: a generative model for seismic events from cotopaxi volcano (Grijalva et al. 2021)	
Used model	Deep convolutional generative adversarial networks (DCGAN)	
Article no.	4*	Based on considerations such as imbalanced volcano datasets and atmospheric noise existence, a CNN model is developed and trained using synthetic InSAR images. A total number of 30,000, 500 by 500 pixels, synthetic images are generated utilizing existing models that are indicative of seismologically related deformations, stratified atmospheric artifacts and turbulent atmospheric artifacts The model is then trained using real data, synthetic data, and a combination of mentioned. Based on the positive predictive value results, the best performance achieved 82%. Finally, it is concluded that training the CNN model with synthetic examples can improve the ability to detect volcano deformations in InSAR satellite images
Article title	A deep learning approach to detecting volcano deformation from satellite imagery using synthetic datasets (Anantrasirichai et al. 2019)	
Used model	CNN	
Article no.	5	An LSTM architecture specifically adapted for eruption prediction is proposed. the logic behind this is based on the assumption that some specific events trigger future eruptions and to take the advantage of the mentioned assumption, an attention layer is deployed for locating and prioritizing the corresponding information in the feature space. The model is trained using a database prepared, by merging 60 s sampling interval records, related to the Sakurajima volcano activities from 2009 to 2016. The sliding step is set to one hour, while the sliding window is considered two hours. The model is
Article title	Can eruptions be predicted? short-term prediction of volcanic eruptions via attention-based long short-term memory (Le et al. 2020)	
Used model	LSTM	

(continued)

Table 7 (continued)

		stated to be more effective than traditional methods based on the quantitative comparisons performed by the authors in the research
Article no.	6	A transfer learning approach is proposed for event classification, in motivation overcoming problems arising from incomplete and (or) limited data situations. A CNN architecture is developed alongside an MLP model and used for classifying seismic signals related to activities of the "Volcán de Fuego" volcano in Mexico. Classification is performed over 9332 samples of volcano-tectonic earthquakes, long-period events, volcanic tremors, regional earthquakes, collapses, explosions and volcanic noise. The dataset is split into 75% training and 25% test sets. Different shallow and deep models have been used for performance evaluation. Based on attaining nearly 94% of events correctly classified, the authors noticed good generalization performance and computational time and resources decrees
Article title	Classification of isolated volcano-seismic events based on inductive transfer learning (Titos et al. 2020)	
Used model	CNN	
Article no.	7*	The dependence of the model generalization ability, on the volume of training data has been considered by the authors of this paper. Classification of volcano related seismic events has been investigated, developing a CNN model with an active learning algorithm that selects training data based on diversity, using an ML method. five classes of events were considered for the signal classification, namely, volcano-tectonic, long period, low frequency, hybrid events and tremor events. Nine thousand two hundred six events corresponded to 2012 unrest and eruption of Nevado del Ruiz Volcano (Colombia), and Llaima Volcano (Chile) over the 2010–2016 non-eruptive period, were used for method training, validating and testing by about 66, 16.5 and 16.5% of the data, respectively. The observations reveal that using the active data selection algorithm, resulted in better testing accuracy concerning the model using randomly selected training data
Article title	A deep active learning approach to the automatic classification of volcano-seismic events (Manley et al. 2022)	
Used model	CNN	

(continued)

Table 7 (continued)

Article no.	8*	The volcanic ash particles have been classified into four classes using a CNN model. The model's targets are the probabilities corresponding to each class, to have a quantitative representation of any possible mixture of four basal particle shapes. A set consisting of 15,000 ash images including the four ash shapes, namely vesicular, blocky, elongated and rounded ashes, is used in the research. The developed model has had an accuracy of over 90%, and it is concluded that the results are consistent with the regarded eruptions mechanism
Article title	Classification of volcanic ash particles using a convolutional neural network and probability (Shoji et al. 2018)	
Used model	CNN	

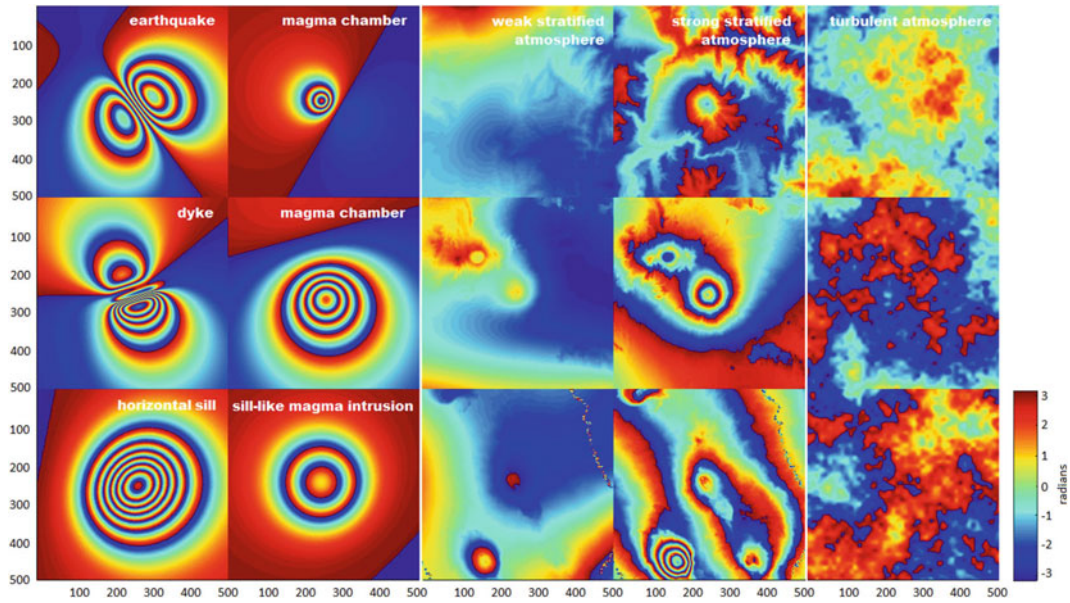


Fig. 24 Some synthetic 500 * 500 pixels InSAR images used for CNN model training. Starting from the first two leftmost Columns, they show different types of seismologically related deformations. Columns 3 and 4 show weak and strong stratified atmosphere and Column 5 shows turbulent atmospheres (Anantrasirichai et al. 2019)

Sentinel-1 radar mission is also used in the research for training and also evaluating sections. Descriptions for the Mentioned data is provided in (Anantrasirichai et al. 2018). Since the real-world interferometers may be, results of a mixed situation (deformation plus atmospheric criteria), as synthetic inputs, a weighted composition of the three specified classes is used. (Fig. 25).

The designed CNN model contains five convolutional layers followed by three fully connected layers. Since two kinds of data are fed to the model, the workflow has been adapted for better performance achievement (authors have used a transfer learning strategy for faster optimization). At the first step, using synthetic images, binary classification is performed as "deformation" and

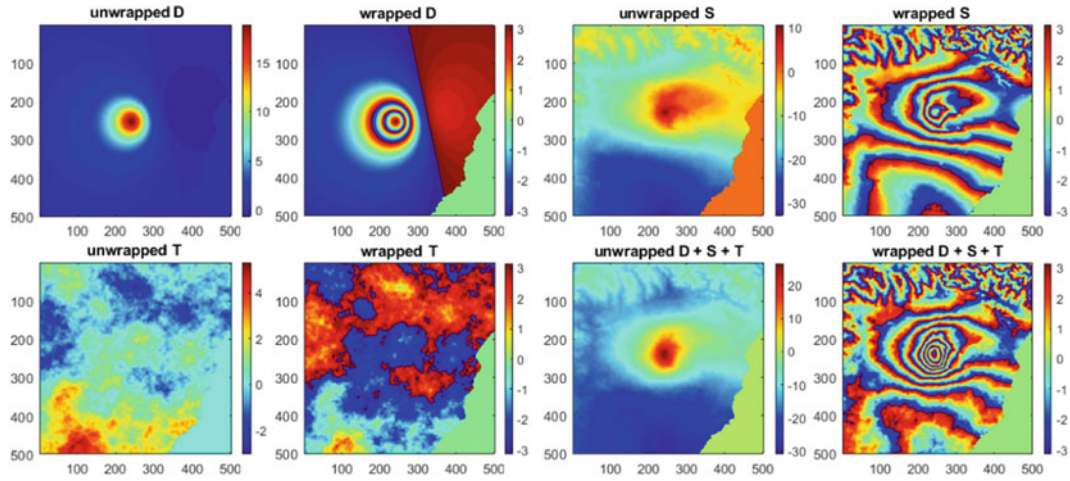


Fig. 25 Combining synthetic unwrapped deformation (D) with stratified atmosphere (S) and turbulent atmosphere (T) as an input for model training (Anantrasirichai et al. 2019)

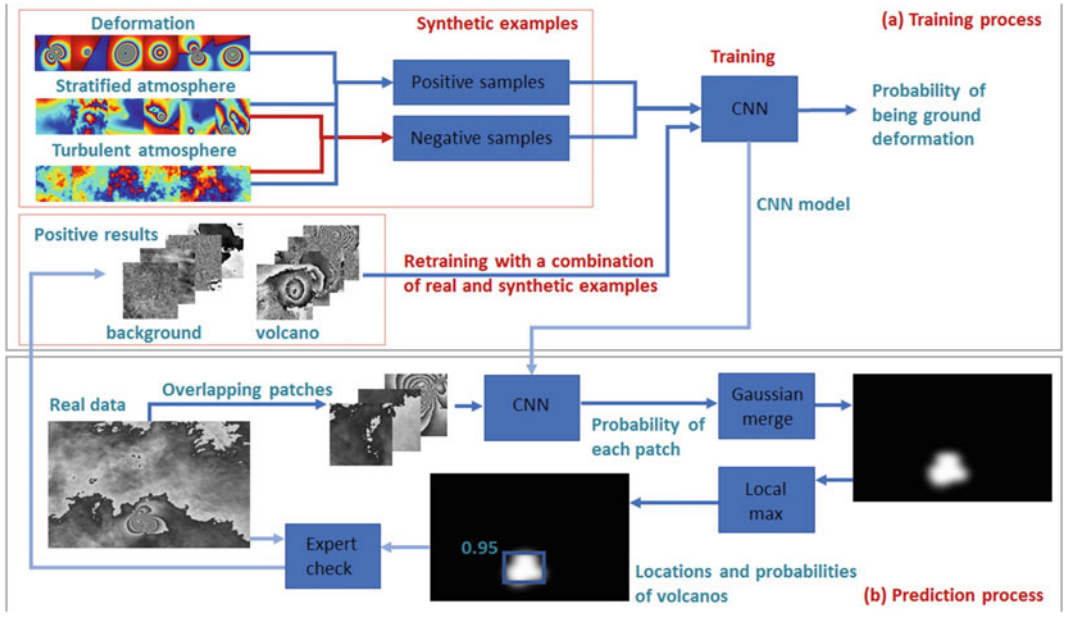


Fig. 26 **a** Training and **b** prediction process used in the proposed framework for deformation identification (Anantrasirichai et al. 2019)

"non deformation" classes. The initial model is then utilized for change detection that makes prediction possible and there so is named the "prediction" part of the framework. After supervising the results by an expert, the model is trained again for final classification. The schematic flow of the algorithm is provided in Fig. 26.

Training the model is performed using 100 samples as batch size and the max epoch is considered as 50. real data, synthetic data, and a combination of mentioned is used for training models in different experiments. Different real datasets are used for evaluation, and the best result is achieved based on a positive predictive value index which is 82%.

Finally, it is concluded that training the CNN model with a mixture of real and synthetic examples can improve the ability to detect volcano deformations using remote sensing datasets such as InSAR satellite images. Detailed and comprehensive discussions are provided especially for the approaches such as "sources of errors", "Atmospheric correction" and "proportion of training data" that could provide insights for future researches.

4.2 Automatic Classification of Volcano-Seismic Signals Using Active Deep Learning to Overcome the Case of Small Training Datasets

Since, the effective usage of a deep learning model, depends on the large amounts of training data, Manley et al. (2022) presented a method aiming mainly, to improve the performance of the models that are trained, with relatively smaller sized training data which uses active learning, instead of a random selection of training pairs in supervised methods. The problem of automatic classification of volcanic data is investigated in the research. Volcano-seismic events are targeted as five classes namely: Volcano-Tectonic (VT), Long Period (LP), Low Frequency (LF), hybrid events (HB) and tremor events (TR). Examples corresponding to each class is shown in Fig. 27.

The presented method utilizes diversity-based active learning proposed by Sener and Savarese (2017) and named KCenterGreedy method. Using distance measuring function, each new sample is selected after it is proven to be the most dissimilar element, to the currently gathered data samples. The procedure of active learning alongside the workflow of the method is illustrated in Fig. 28.

The applied CNN model is developed initially by Canário et al. (2020). The SeismicNet architecture (Fig. 28a) consists of successive combinations of 1D convolutional layers followed by

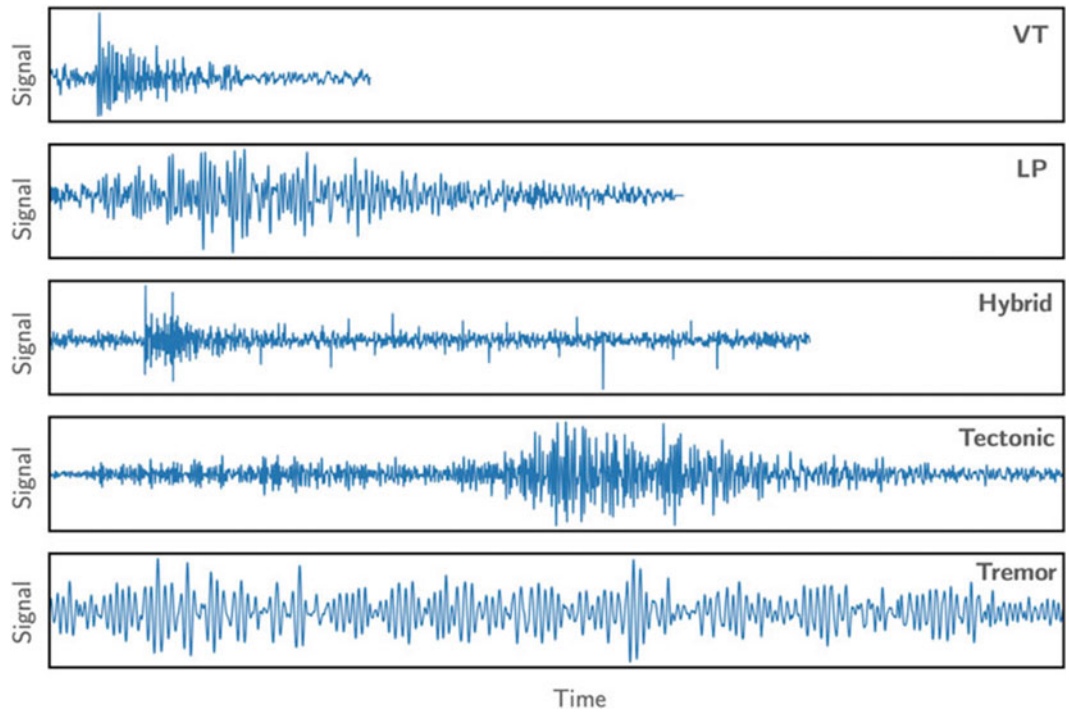


Fig. 27 Examples of waveforms corresponded to each class (Manley et al. 2022)

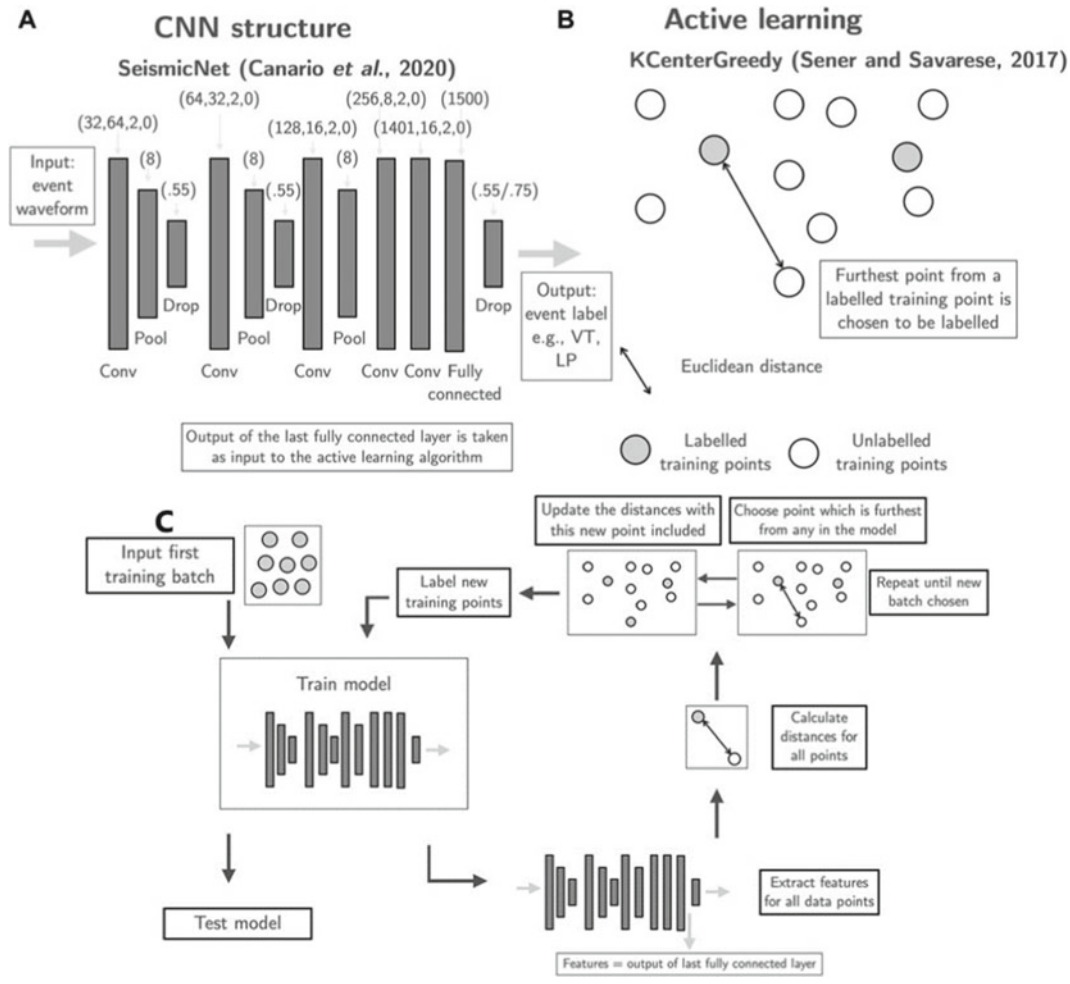


Fig. 28 **a** SeismicNet architecture. **b** Schematic illustration of the KCenterGreedy algorithm, grey points indicate samples within the training dataset and white points represent unlabeled samples. Arrows are representing the minimum distance in the feature space. **c** Active learning included in the model evaluation experiment (Manley et al. 2022)

Table 8 Hyper-parameters considered for CNN model training

Parameter	Value/range
Optimization algorithm	Stochastic gradient descent (SGD)
Loss function	Categorical cross entropy
Learning rate	Not specified
Epoch	150
Batch size	16

max pooling, dropout, and Relu activation function. Finally, two fully connected layers are applied prior to the output layer. Hyper-parameters considered for model training are specified in Table 8.

Experiments are performed using a combination of two datasets: the first one, is a set of 5614 events, recorded from the 2012 unrest and eruption of Nevado del Ruiz volcano (Colombia) and the second data, consists of 3592 events

Table 9 Specifications of the ash particles and imagery of them

Item	Specification
Regarding volcano type	Monogenetic basaltic volcano
Eruption mechanism	Magmatic, phreatomagmatic, and rootless
Grain sizes (diameter size)	125–250 μm
Imagery illumination system	Diascopic with automatic light calibration
Final image dimensions	50 by 50 pixles
Background threshold separation	80 form the range of [0, 255]

recorded between 2010 and 2016 during non-eruptive period from Llaima Volcano (Chile). For training, validating and testing subsets, portions of about 66, 16.5 and 16.5% of the data are used.

Based on the detailed discussion over the results, it is concluded that the active learning approach achieved superior performance than models trained with randomly selected samples.

4.3 Probabilistic Shape Classification of the Volcanic Ash Using Convolutional Neural Networks

As the shape of volcanic ash is the result of the eruption mechanism, in the valuable research presented by Shoji et al. (2018), using a CNN model, a method has been presented aiming to overcome the difficulty of visual classification of the ash particle classification. One of the main considerations here is to classify the particles in a manner that supports the identification of the complicated shapes that are supposed to belong to a mixture of shapes.

A dataset including 15,000 ash particle images captured by Morphologi G3STM Instruments is used in the study resulting from three eruption mechanisms, namely magmatic, rootless, and phreatomagmatic, related to three volcanic zones (Izu Paninsula and Miyakejima Island both in Japan and Myvatn, Iceland). Details about particles and their imagery specifications are provided in Table 9.

The four classes considered in the research are illustrated in Fig. 29. The rectangular-shaped particles with relatively nearly right-angle edges are regarded as "Blocky" particles. The irregular shape samples are categorized as "Vesicular" particles.

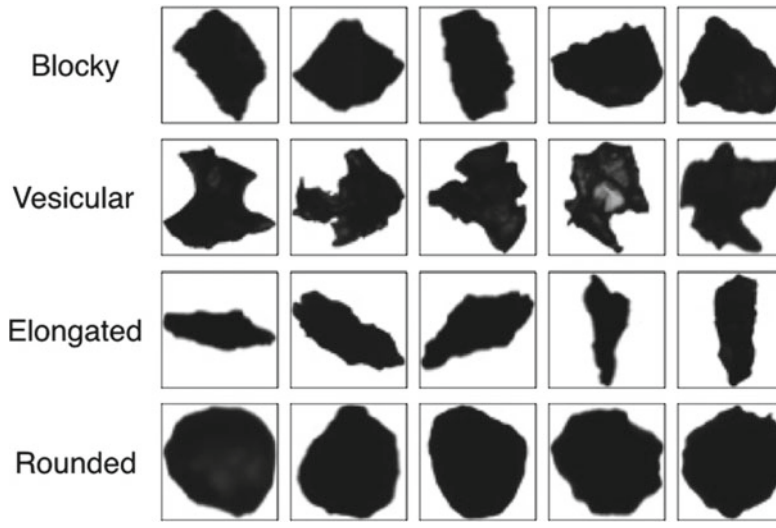


Fig. 29 Examples of four classes used for ash particle classification (Shoji et al. 2018)

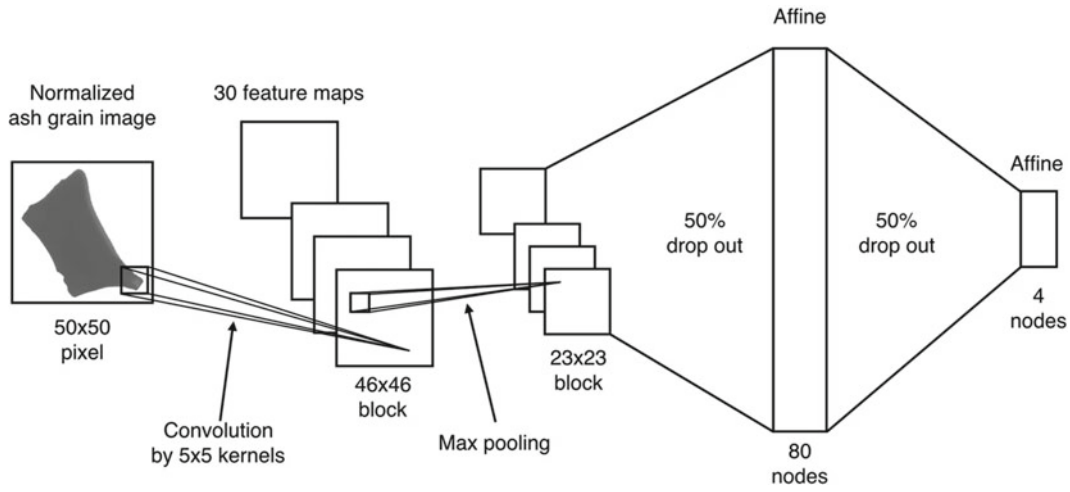


Fig. 30 The CNN model used for ash particle classification (Shoji et al. 2018)

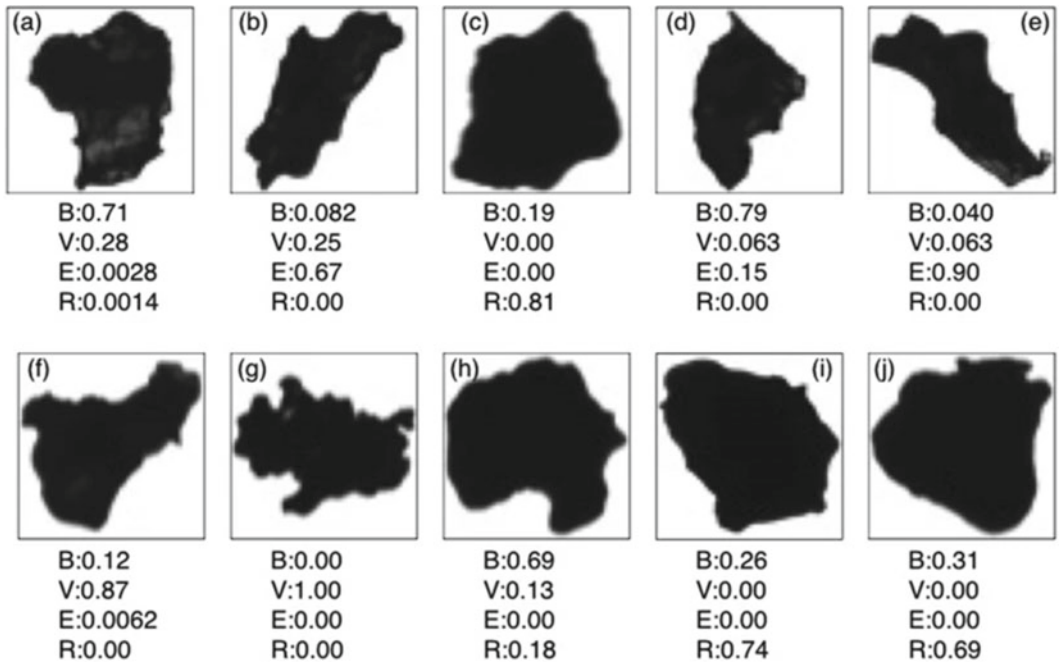


Fig. 31 Examples showing the results of generalizing the trained model, to the test data samples. The corresponding class probabilities for each basal shape are: B for the blocky, V for the vesicular, E for the elongated and R for the rounded class (Shoji et al. 2018)

"Elongated" particles correspond to the relatively long and thin samples, and finally, "Rounded" particles are supposed to have a rounded shape. The authors provide a comprehensive study about the process of formation of each particle in the paper.

The CNN model used in the research is illustrated in Fig. 30. The input images are fed to the model after normalizing. Also, data augmentation and 50% drop-out blocks before each fully connected layer are used based on the

consideration to avoid an overfitting situation. The reported hyperparameters used in model training include Optimization Algorithm (ADAM), epoch (100) and batch size (200).

A set of test images were used for the method evaluation resulting in approximately 92% as classification accuracy. Examples of mentioned generalization are plotted in Fig. 31 where, the possibilities related to each class is also mentioned below each sample. In a greater portion of samples like Fig. 30a, b, and i, two prominent basal shapes, seem to be detected, while, in other cases, three and four and rarely one classes has been identified in the mixture. A detailed discussion regarding the petrophysical justifications for the distribution of the probabilities, is provided in the paper.

Finally, it is concluded that identified averaged probabilities and relating clustering of the samples, is consistent with the eruption type. Furthermore, It is stated that the mixing ratio resulting from generalizing the CNN model can be utilized to quantitatively classify complex shapes of the ash particles, which may lead to a new taxonomy method.

References

Aki K, Richards PG (2002) Quantitative seismology, 2nd edn. University Science Books, Sausalito. http://www.worldcat.org/isbn/0935702962

Anantrasirichai N, Biggs J, Albino F, Bull D (2019) A deep learning approach to detecting volcano deformation from satellite imagery using synthetic datasets. Remote Sens Environ. https://doi.org/10.1016/j.rse.2019.04.032

Anantrasirichai N, Biggs J, Albino F, Hill P, Bull D (2018) Application of machine learning to classification of volcanic deformation in routinely generated InSAR data. J Geophys Res Solid Earth. https://doi.org/10.1029/2018JB015911

Banna MdH, Taher KA, Kaiser MS, Mahmud M, Rahman MdS, Hosen ASMS, Cho GH (2020) Application of artificial intelligence in predicting earthquakes: state-of-the-art and future challenges. IEEE Access 8:192880–192923. https://doi.org/10.1109/ACCESS.2020.3029859

Bonheur S, Štern D, Payer C, Pienn M, Olschewski H, Urschler M (2019) Matwo-CapsNet: a multi-label semantic segmentation capsules network, pp 664–672. https://doi.org/10.1007/978-3-030-32254-0_74

Bueno A, Benitez C, de Angelis S, Diaz Moreno A, Ibanez JM (2020) Volcano-Seismic Transfer Learning and Uncertainty Quantification with Bayesian Neural Networks. IEEE Trans Geosci Remote Sens 58(2):892–902. https://doi.org/10.1109/TGRS.2019.2941494

Canário JP, Mello R, Curilem M, Huenupan F, Rios R (2020) In-depth comparison of deep artificial neural network architectures on seismic events classification. J Volcanol Geoth Res. https://doi.org/10.1016/j.jvolgeores.2020.106881

Cheng X, Liu Q, Li P, Liu Y (2019) Inverting Rayleigh surface wave velocities for crustal thickness in eastern Tibet and the western Yangtze craton based on deep learning neural networks. Nonlinear Process Geophys 26(2):61–71. https://doi.org/10.5194/npg-26-61-2019

Crotwell HP, Owens TJ, Ritsema J (1999) The TauP Toolkit: flexible seismic travel-time and ray-path utilities. Seismol Res Lett 70(2):154–160. https://doi.org/10.1785/gssrl.70.2.154

del Rosso MP, Sebastianelli A, Spiller D, Mathieu PP, Ullo SL (2021) On-board volcanic eruption detection through CNNs and satellite multispectral imagery. Remote Sens 13(17). https://doi.org/10.3390/rs13173479

Derakhshani A, Foruzan AH (2019) Predicting the principal strong ground motion parameters: a deep learning approach. Appl Soft Comput J 80:192–201. https://doi.org/10.1016/j.asoc.2019.03.029

Florez MA, Caporale M, Buabthong P, Ross ZE, Asimaki D, Meier M-A (2020, November 17) Data-driven accelerogram synthesis using deep generative models. AGU2020 Fall Meeting. http://arxiv.org/abs/2011.09038

Grijalva F, Ramos W, Perez N, Benitez D, Lara R, Ruiz M (2021) ESeismic-GAN: a generative model for seismic events from Cotopaxi volcano. IEEE J Select Top Appl Earth Observ Remote Sens 14:7111–7120. https://doi.org/10.1109/JSTARS.2021.3095270

Hu J, Qiu H, Zhang H, Ben-Zion Y (2020) Using deep learning to derive shear-wave velocity models from surface-wave dispersion data. Seismol Res Lett 91(3):1738–1751. https://doi.org/10.1785/0220190222

Jozinović D, Lomax A, Štajduhar I, Michelini A (2021) Rapid prediction of earthquake ground shaking intensity using raw waveform data and a convolutional neural network. Geophys J Int 222(2):1379–1389. https://doi.org/10.1093/GJI/GGAA233

Kavianpour P, Kavianpour M, Jahani E, Ramezani A (2021) A CNN-BiLSTM model with attention mechanism for earthquake prediction. http://arxiv.org/abs/2112.13444

Kossobokov VG, Romashkova LL, Panza GF, Peresan A (2002) Stabilizing intermediate-term medium-range earthquake predictions. In: JSEE: Summer and Fall, vol 4, no 3

Kuang W, Yuan C, Zhang J (2021) Real-time determination of earthquake focal mechanism via deep learning. Nat Commun 12(1). https://doi.org/10.1038/s41467-021-21670-x

Le H, Murata T, Iguchi M (2020) Can eruptions be predicted? Short-term prediction of volcanic eruptions via attention-based long short-term memory. Proc AAAI Conf Artif Intell 34(08):13320–13325. https://doi.org/10.1609/aaai.v34i08.7043

Manley GF, Mather TA, Pyle DM, Clifton DA, Rodgers M, Thompson G, Londoño JM (2022) A deep active learning approach to the automatic classification of volcano-seismic events. Front Earth Sci. https://doi.org/10.3389/feart.2022.807926

Mousavi SM, Beroza GC (2019) Bayesian-deep-learning estimation of earthquake location from single-station observations. IEEE Trans Geosci Remote Sens. https://doi.org/10.1109/TGRS.2020.2988770

Mousavi SM, Beroza GC (2020) A machine-learning approach for earthquake magnitude estimation. Geophys Res Lett 47(1). https://doi.org/10.1029/2019GL085976

Mousavi SM, Ellsworth WL, Zhu W, Chuang LY, Beroza GC (2020) Earthquake transformer—an attentive deep-learning model for simultaneous earthquake detection and phase picking. Nat Commun 11(1). https://doi.org/10.1038/s41467-020-17591-w

Peng P, He Z, Wang L, Jiang Y (2020) Microseismic records classification using capsule network with limited training samples in underground mining. Sci Rep 10(1). https://doi.org/10.1038/s41598-020-70916-z

Perol T, Gharbi M, Denolle M (2018) Convolutional neural network for earthquake detection and location. Sci Adv 4(2). https://www.science.org

Pu Y, Chen J, Apel DB (2021) Deep and confident prediction for a laboratory earthquake. Neural Comput Appl 33(18):11691–11701. https://doi.org/10.1007/s00521-021-05872-4

Li R, Lu X, Li S, Yang H, Qiu J, Zhang L (2020) DLEP: a deep learning model for earthquake prediction. Int Joint Conf Neural Netw (IJCNN)

Sener O, Savarese S (2017) Active learning for convolutional neural networks: a core-set approach. http://arxiv.org/abs/1708.00489

Shelhamer E, Long J, Darrell T (2017) Fully convolutional networks for semantic segmentation. IEEE Trans Pattern Anal Mach Intell 39(4):640–651. https://doi.org/10.1109/TPAMI.2016.2572683

Shoji D, Noguchi R, Otsuki S, Hino H (2018) Classification of volcanic ash particles using a convolutional neural network and probability. Sci Rep 8(1):8111. https://doi.org/10.1038/s41598-018-26200-2

Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556

Stepnov A, Chernykh V, Konovalov A (2021) The seismo-performer: a novel machine learning approach for general and efficient seismic phase recognition from local earthquakes in real time. Sensors 21(18). https://doi.org/10.3390/s21186290

Sugiyama D, Tsuboi S, Yukutake Y (2021) Application of deep learning-based neural networks using theoretical seismograms as training data for locating earthquakes in the Hakone volcanic region, Japan. Earth Planets Space 73(1). https://doi.org/10.1186/s40623-021-01461-w

Titos M, Bueno A, García L, Benítez C, Segura JC (2020) Classification of isolated volcano-seismic events based on inductive transfer learning. IEEE Geosci Remote Sens Lett 17(5):869–873. https://doi.org/10.1109/LGRS.2019.2931063

Uchide T (2020) Focal mechanisms of small earthquakes beneath the Japanese islands based on first-motion polarities picked using deep learning. Geophys J Int 223 (3):1658–1671. https://doi.org/10.1093/gji/ggaa401

Wang J, Xiao Z, Liu C, Zhao D, Yao Z (2019) Deep learning for picking seismic arrival times. J Geophys Res Solid Earth 124(7):6612–6624. https://doi.org/10.1029/2019JB017536

Yousefzadeh M, Hosseini SA, Farnaghi M (2021) Spatiotemporally explicit earthquake prediction using deep neural network. Soil Dyn Earthq Eng. https://doi.org/10.1016/j.soildyn.2021.106663

Zhao D, Yanada T, Hasegawa A, Umino N, Wei W (2012) Imaging the subducting slabs and mantle upwelling under the Japan Islands. Geophys J Int 190(2):816–828. https://doi.org/10.1111/j.1365-246X.2012.05550.x

Zhu J, Li S, Song J, Wang Y (2021) Magnitude estimation for earthquake early warning using a deep convolutional neural network. Front Earth Sci. https://doi.org/10.3389/feart.2021.653226

Zhu L, Helmberger D (1996) Advancement in source estimation techniques using broadband regional seismograms. Bull Seismol Soc Am 86(5):1634–1641. https://doi.org/10.1785/BSSA0860051634

Zhu L, Rivera LA (2002) A note on the dynamic and static displacements from a point source in multilayered media. Geophys J Int 148(3):619–627. https://doi.org/10.1046/j.1365-246X.2002.01610.x

Evolutionary Algorithms with Focus on Genetic Algorithm

Abstract

In this chapter, an important group of intelligent optimization methods called evolutionary algorithms is presented, and among these methods, the genetic algorithm (GA), which is one of the most common evolutionary algorithms, is specifically focused on so that the reader, in addition to basic concepts, in details get acquainted with the GA method.

1 Evolutionary Computation

1.1 Introduction

Today, one of the most important fields of research in biological computation is the development of search methods based on the principles of natural evolution. Of course, most book readers are probably familiar with the basic concepts of evolution through books and/or educational experiences. In particular, Darwin's theory of "natural selection" expressed by Charles Darwin is very popular. According to this theory, all plants and animals in the world today are the result of millions of years of adaptation to environmental requirements. In fact, at any given time, different organisms have lived side by side and competed for available natural resources. In this way, more successful organisms in obtaining natural resources were able to reproduce in greater numbers in the future, and organisms that had less capacity for any reason were either extinct or few remained in nature.

Evolutionary computations are abstractly inspired by the basic concepts of natural evolution in search of optimal solutions to various problems.

In search algorithms, there are solutions to the problem and the goal is to find a solution in a certain time. In small search spaces, each of these solutions can be reviewed in a reasonable time and the optimal solution can be selected. This type of search is called a complete search, but as the search space expands, it quickly becomes impractical.

Conventional search algorithms, including random and heuristic algorithms, search the search space at any time step in the hope of finding an optimal solution. The key aspect that distinguishes evolutionary search algorithms from conventional algorithms is that evolutionary algorithms are population-based.

Evolutionary algorithms perform an efficient direct search by successively adapting generations to specific conditions. Therefore, these methods have better performance compared to random and exploratory methods.

In this chapter, Literature review of evolutionary computations are briefly discussed and the process of artificial evolution is presented. The basic principles of evolution, which are based on Darwin theory, are also described in

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

A. Hajian et al., *Intelligent Methods with Applications in Volcanology and Seismology*, Advances in Volcanology, https://doi.org/10.1007/978-3-031-15432-4_6

this chapter. In addition to the above, a brief overview of genetics and optimization technique is presented.

1.2 Annals of Evolutionary Computing: A Brief Literature Review

Natural evolution is a historical process, in other words, any entity that changes over time is a historical entity. The nature of evolution is changing. Evolution is a dynamic two-stage process of "change" and "random selection" that creates fixed changes in individuals in a population based on dynamic environment and requirements.

Individuals in the population gradually adapt to their current environment through anthology (excerpting), an adaptation that allows certain individuals to survive through inheritance. Thus, adaptation is a historical sequence of successive events.

Here is an interesting example. Suppose a teacher conveys a message to one of the students and the students whisper the message to each other. Every student hears only what the last student whispered in his/her ear. The message changes slightly at every step.

Finally, the last student expresses the message he/she heard. This message is different from the original message. Therefore, it can be said that history, like science, needs to be constantly revised because a wrong statement from a writer eventually becomes a myth and is accepted without any questions and passed down from one generation to the next generation.

In the last seventy years, computers have been widely used in various sciences. The application of Darwinian principles stems from the need to solve automation problems. In the 1960s, three different interpretations of the idea were developed in three different places. *Evolutionary programming* was first introduced in the United States by Fogel (1962a, b). At the same time, Holland(1962) called this method the *genetic algorithm*, and in Germany, Rechenberg (1965) and Schwefel (1968) proposed *evolutionary strategies*. These fields were developed separately over fifteen years. In the nineties, everyone noticed different representations of a technology, and that was nothing but evolutionary computing. Therefore, all four fields of *evolutionary programming, evolutionary strategies, genetic algorithms and genetic programming* can be considered as a subset of evolutionary computing (Fogel 2012).

1.3 Biological and Artificial Evolutionary

Evolutionary computing (EC) is based on the biological and natural evolution of living things in nature. To describe three different fields of EC, in this section we will explain some of the terms used in EC with a brief look at the key concepts of natural evolution that are the main base of EC algorithms. Some of the terms which borrowed from nature for use in EC are listed in Table 1. In this table the terms of biological evolutionary and their corresponding meaning in EC are briefed.

2 Genetic Algorithm

2.1 Introduction to Natural Genetics

The body of every living thing is made up of cells, and every cell is made up of chromosomes. Chromosomes are also made up of DNA strands. Chromosomes are also made up and each block of DNA is called a gene, and each gene is made up of a specific and unique protein. A set of genes is called a genome. Amongst the terms that has been entered from biology to genetic

Table 1 Terms of biological evolutionary and their corresponding in EC

Biological term	Meaning in EC
Chromosome	String of symbols
Population	A set of chromosomes
Gene	A feature

algorithms the following terms are necessary to be mentioned:

2.2 Crossover

The production of a new chromosome by a combination of parent genes is called crossover.

2.3 Mutation

Sudden changes in DNA are called mutations.

2.4 Fitness

The success of a living thing in creating life and forming a new generation is called fitness. Of course, the above subject may seem a little incomprehensible at first glance, but in the following, we will explain more about them and also their application in genetic algorithm.

Figure 1 shows a biological view of cell nucleus and Fig. 2 presents detailed anatomy of the animal cell. In Table 2. genetic algorithm terminology with each equivalent mathematical programming is listed.

3 Fundamentals of Genetic Algorithms

In this section we briefly explain encoding, cross over and mutation with some examples.

3.1 Encoding

Each member of the population, which is an approximation of the final answer, is encoded as strings of letters or numbers. These strings are called *chromosomes*. The most common way to display is binary digits. Other modes such as the use of three digits, real numbers and integers are also used. For example, a chromosome with two variables **a** and **b** (2 genes) is represented by the structure depicted in Fig. 3.

Variable "a" is displayed with the first 10 cells on the right and "b" with the remaining 15 cells. Note that generally, the number of digits is assigned due to the level of accuracy or range of the decision variable.

An example of real coding is shown in Fig. 4. Assume a multilayer model of the subsurface ground where each layer may have no fault (code: 0) or a fault fracture with dip angle (0°–90° with 10° resolution). In order to remind dip-angle concept for the readers a pictorial schematic is shown in Fig. 5. To code the dip angle of probable fault located in each layer of the model, each angle is encoded with a number while one or more layers might have no fault which means the fracture in the related layer(s) is not categorized as a fault. It's obviously that code '1' that refers to 0° dip angle means a thrust fault and code '10' which refers to 90° dip angles is meaning a strike fault (Fig. 6).

3.2 Fitness

The values on the chromosomes alone have a special meaning, but they must be decoded into real values in order to have meaning and result as decision variables. After the chromosomes have been decoded, efficiency can be calculated by fitting each individual into the population. Fitness is a relative scale that indicates a person's eligibility to produce the next generation. In nature, fitness is equivalent to one's ability to survive. The objective function plays a decisive role in determining the fitness of individuals.

3.3 Cross Over

During crossover, with the help of the basic information obtained from the objective function, the fit of each individual is determined. These values are used in the selection process to lead it to select the right people.

The higher the fitness of the individual in relation to the population, the more chance to be selected. The lower the relative fitness, the less chance to be selected for next-generation production.

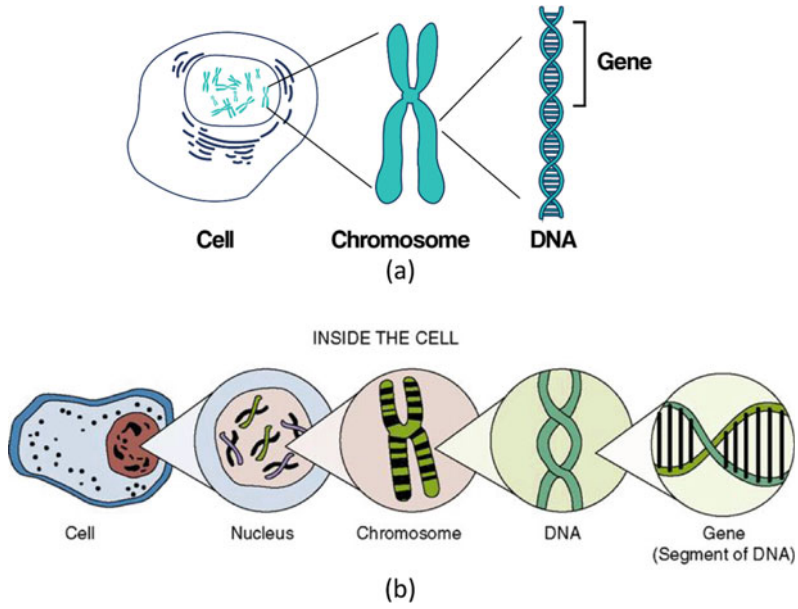


Fig. 1 Schematic view of an animal cell with chromosome, DNA and Gene; **a** linear presentation, **b** waterfall representation (references **a** https://www.pngitem.com/middle/hRoTRhb_transparent-chromosome-clipart-chromosomes-and-genes-hd-png/, **b** https://www.civilsdaily.com/biotechnology-basics-of-cell-nucleus-chromosomes-dna-genes-etc/)

Once the fit of all members of the population has been determined, each with a probability commensurate with their fit can be selected to produce the next generation. The act of amplification in a genetic algorithm is used to exchange genetic information between a pair or more individuals. The simplest type of amplification is a one-point cross-over.

Consider the two strings in Fig. 7, If an integer is chosen from one or the number of digits in a string minus one, and we change the information of the two strings on either side of these two points, two new strings are created, which are called children. For example, if we choose the number 6 for the two strings of Fig. 7, the result is a one-point intersection of Fig. 7. Figure 8 shows the cross over in the actual coding.

3.4 Mutation

This operator does not necessarily apply to all strings in a population, but assigns a probability to apply to a pair of strings. After this step, with a new probability, the mutation operator is applied to the generated strands. In mutation, each individual can change according to the laws of probability. In a binary representation of strings, a mutation means changing the value of one of the string cells from zero to one or inversely from one to zero (Fig. 9).

Apart from the two cross-over and mutation operators that are used in all genetic algorithms, there are some other operators that are used in special problems. Among them, we can name the addition operator with the deletion operator and also the swap operator.

After the steps of amplification and mutation, the chromosomes are decoded and the value of the objective function of each is calculated. Then a fit is assigned to each. Now, if necessary, the selection, crossover, etc. steps will be performed again. During this process, the average efficiency of the response population is expected to be increased. The algorithm ends when a specific goal is met. For example, if the personal number of generations is created, the deviation of the average efficiency of individuals reaches a specified value, or reach a certain point in the search space.

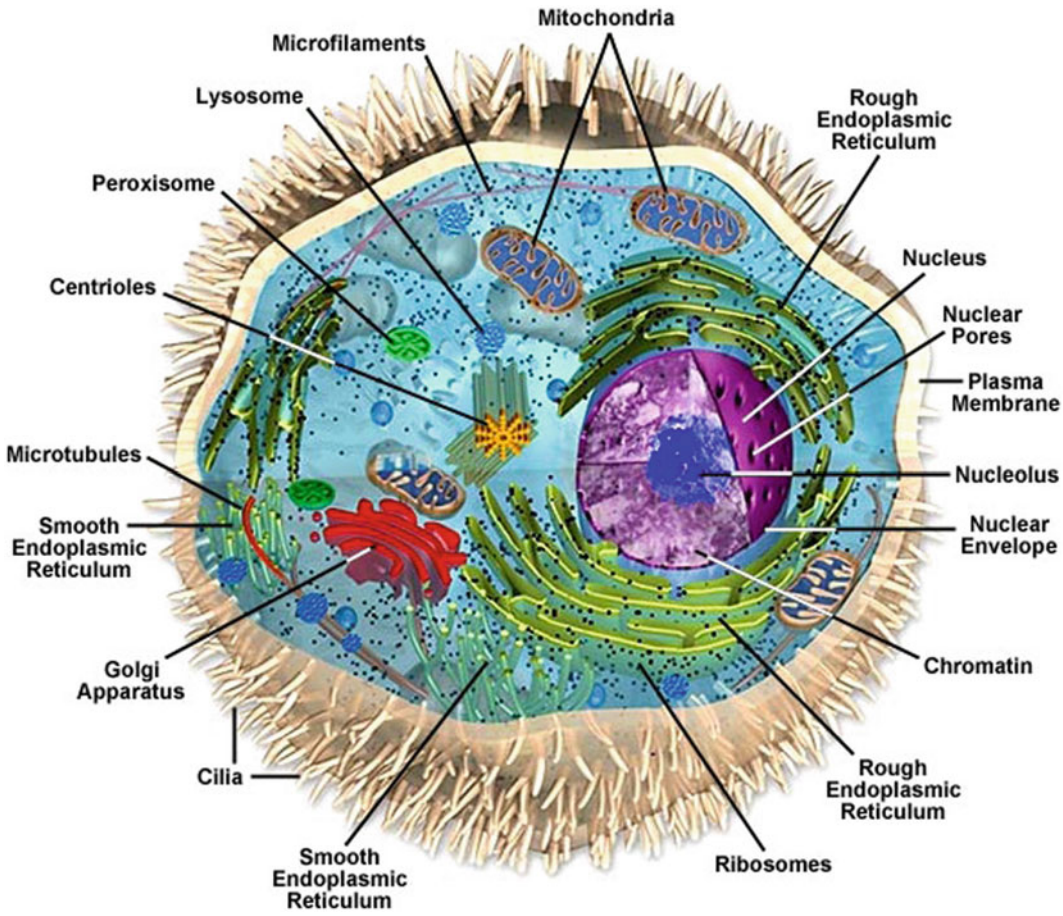


Fig. 2 Anatomy of the animal cell (reference: https://micro.magnet.fsu.edu/cells/animals/animalmodel.html)

Table 2 Genetic algorithm terminology (Shopova and Vaklieva 2006)

Genetic terminology	Mathematical programming equivalent
Generation	Iteration
Chromosome or individual or chromosome genotype	Coded vector of control variables
Chromosome phenotype	Vector of real values of control variables
Gene—part of chromosome which represents given feature	Coded particular variable
Morphogenesis or growth function—transforms chromosome genotype in chromosome phenotype	Decoding function
Population set of parent chromosomes	Set of vectors of control variables
Objective function	Quality model characteristic for optimization
Fitness function	Normalized objective function at iteration t

variable **a**(gene1)													variable **b**(gene2)									
1	0	0	1	1	0	0	1	0	1	1	0	0	1	0	1	0	1	1	1	1	0	1	0	0

Fig. 3 A sample representation of a 2-gene chromosome with binary digits

Fig. 4 Integer coding of fault dip angles

0=no fault

1=fault with dip angle 0^0 (thrust fault)

2=fault with dip angle 10^0

3= fault with dip angle 20^0

4= fault with dip angle 30^0

5= fault with dip angle 40^0

6= fault with dip angle 50^0

7= fault with dip angle 60^0

8= fault with dip angle 70^0

9= fault with dip angle 80^0

10= fault with dip angle 90^0(strike-slip fault)

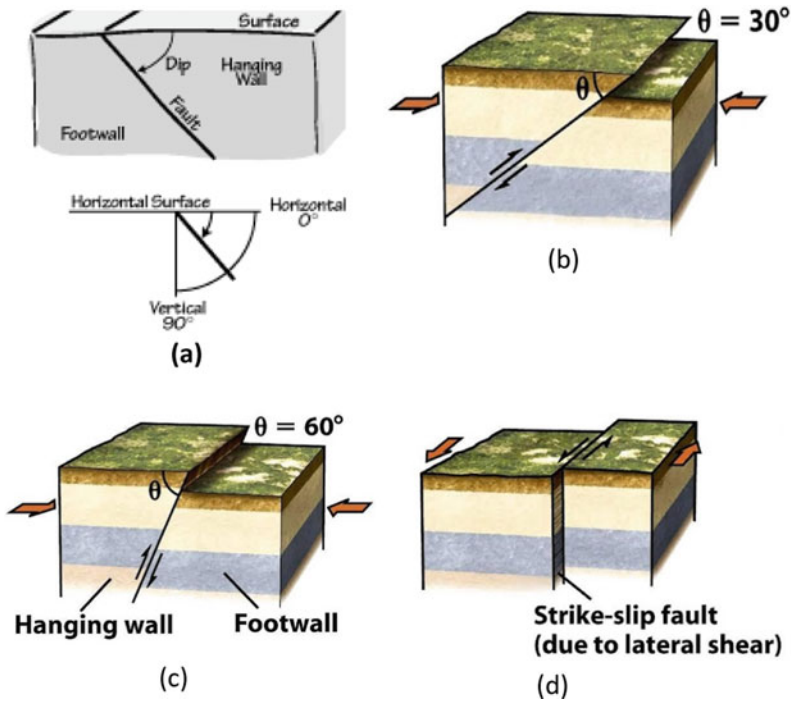


Fig. 5 **a** Dip angle of a fault. **b** A reverse fault with dip angle of 60° and compression mechanism. **c** A thrust fault with dip angle of 30°. **d** A strike-slip fault; note that revers faults have dip of greater than 45° while thrust faults mostly have dip angles less than 45° (references; **a** https://earthquake.usgs.gov/learn/glossary/?term=dip, **b**–**d** https://sites.google.com/site/ldgeohazards/what-are-earthquakes/fault-systems)

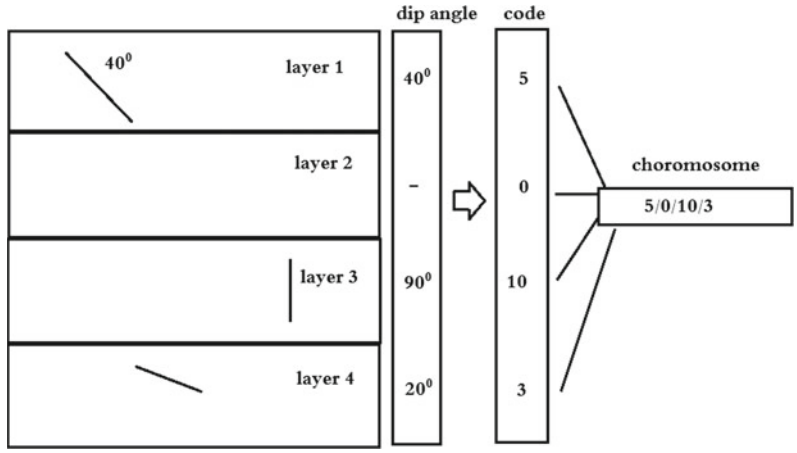


Fig. 6 A sample of 2-D model of the subsurface ground with four layer and three faults with different dip angles encoded reference to Fig. 4 and the related chromosome

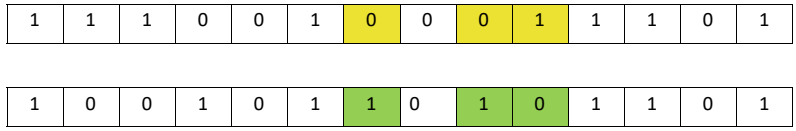


Fig. 7 Two chromosomes before crossing over (parents)

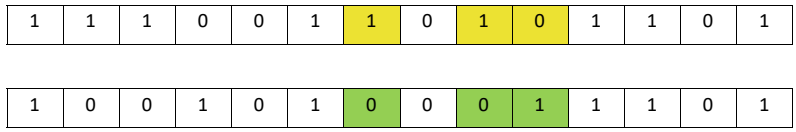


Fig. 8 Two chromosomes after crossing over (children)

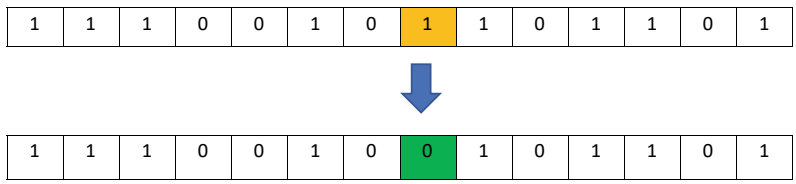


Fig. 9 Chromosome after mutation

4 How to Run Each Parts of the Genetic Algorithm?

In this section, the simple genetic algorithm described by Goldberg is used to explain the main parts. The virtual pseudo code shown in Fig. 10 contains the important points of a simple genetic algorithm. The population at time "t" is represented by a time-dependent variable "P(t)". This pseudo code describes the important components of a simple genetic algorithm.

```
t=0
initialize P(t)
evaluate P(t)
begin
t=t+1
select P(t) from P(t-1)
reproduce pair in P(t)
evaluate P(t)
end
end
```

Fig. 10 Pseud code of a simple genetic algorithm

4.1 Population Representation and Initializing the Algorithm

The genetic algorithm is applied to a set of answers called populations. Individuals are a population of strings of numbers called chromosomes, which contain information encoded in decision parameters. Typically, a population consists of 30–100 individuals, although up to about 10 individuals are used.

In genetic algorithm, the most common way to represent chromosomes is the form of binary strings. Each decision variable is binary and then chromosomes are created by putting these variables together. Although this is the most widely used method of coding, other methods are being developed, such as representations of real numbers. In the context of some issues, this argument has arisen that the binary representation complicates the nature of the search and, in addition, has other problems. For example, consider the two coded numbers 100000000 and 01111111. The actual values of these two numbers differ by only one unit, but the coded values are very different in appearance. These differences become problematic when applying conventional genetic operators. In other words, a small change in the encoded space does not cause a similar change in the unencoded space. To solve this problem, usually 'Gray codes' are used. A Gray code is an encoding of numbers such that two successive numbers have a single digit difference in only one bit (https://mathworld.wolfram.com/GrayCode.html). In example the representation of the integer value "3" and "4" in gray code are "0010", "0110" so they have only on bit difference while if binary code was used three of the bits would be different (see Table 3 for more details about comparison of Binary coding and Gray coding).

The representation of genes with real numbers is proposed to obtain scores in the optimization of numerical functions. This display method is used because it does not need to be decoded and requires less memory. In this method, because we do not have binary mode conversion, the accuracy of the values is not reduced due to this conversion. Coding with real numbers gives better results for optimizing functions. An important point to be noted in the coding process is the nature of the problem and its constraints. After applying these genetic operations to the chromosomes, new answers are obtained that may not be true of the problem. This happens in many constrained issues. The simplest solution to this problem is to use the penalty function in the target function.

In this way, the fitting of out-of-constraint answers is greatly reduced. As a result, the selection process tends toward the chromosomes that are most true to the constraints. The higher the penalty, the faster the algorithm converges and the higher the probability of reaching the local optimum. For this reason, the amount of penalty in various issues is determined by trial and error, and this is a problem that is taken from the penalty function. In some cases, the resulting chromosomes may not be the answer at all. For example, assume a chromosome that partially represents the gases in the smoke emitted from a volcanic chimney during its eruption. Ninety-nine percent of the gas molecules emitted during a volcanic eruption are water vapor (H_2O), carbon dioxide (CO_2), and sulfur dioxide (SO_2). The remaining one percent is comprised of small amounts of hydrogen sulfide (H_2S), carbon

Table 3 Comparison of binary coding and gray coding

Integer	Binary code	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

monoxide (CO), hydrogen chloride (HCL), hydrogen fluoride (HF), and other minor gas species which are shown in Fig. 11. (https://www.usgs.gov/faqs/what-gases-are-emitted-kilauea-and-other-active-volcanoes).

Hence, it is reasonable to show the composition of an active volcano smoke with three different main gases and use the binary codes "00" as H_2O, "01" as CO_2, and "10" as "SO_2". We see that this property occupies two cells of the chromosome. Now, after applying genetic procedures, the code "11" may be placed, while this type of gas does not exist externally as the main gasses in the volcanic smoke and the chromosome created is not a solution to the problem. In this case, because the target function of the chromosome is unanswered, the penalty function does not work. Maybe you guess that this problem is happened because we assumed only three gases of the emission but really this is not the problem. To ensure you about it, in return, we assume all the seven gasses listed with their binary codes in Table 4, in this state the representation is possible by coding in three bits. Now, if again after cross over or mutation one child is (111) it won't be acceptable as this type of gas is not exist in the list. To solve this problem, this type of chromosome is repaired before going to the next step. In each case, according to its nature, different repair methods are used.

Briefly an encoding system should have the properties below:

1. The conversion between coded and non-coded elements must be a one-to-one conversion.
2. Repair chromosomes that are not the answer.
3. Any point in the response space can be converted to a chromosome.
4. The good qualities of parents can be transmitted to children.
5. Slight changes in coded variables cause small changes in non-coded variables.

After deciding on the coding method of the chromosomes, the initial population should be created. This step is usually done by randomly selecting values within the allowable range.

It should be noted that there are both continuous and discrete variables in the chromosome. When applying operators, if necessary, a constraint for discrete variables must be considered and finally their value can be converted to allowable values.

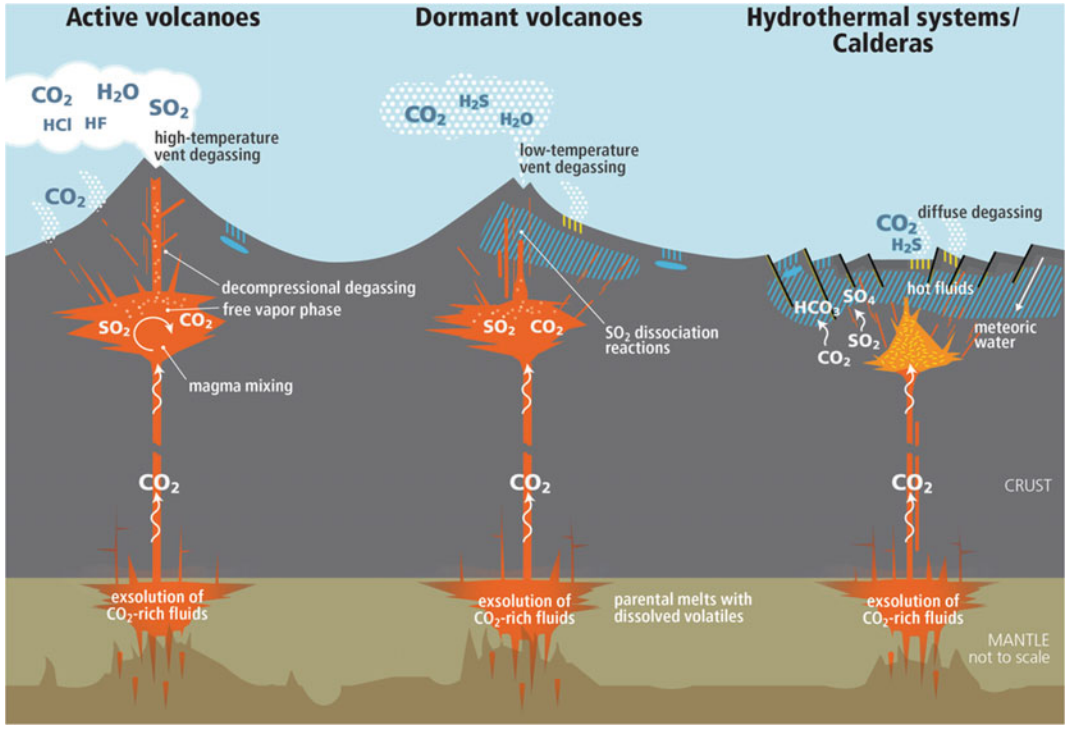


Fig. 11 Sketch showing typical CO_2 emission patterns from volcanic and magmatic systems with other main gases H_2O and SO_2 (by Mice of Mu—Own work, CCBY-SA4.0, https://commons.wikimedia.org/w/index.php?curid=83685527)

Table 4 Encoding of gas types available in volcano smoke

Gas type	Binary code
H_2O	000
CO_2	001
SO_2	010
H_2S	011
CO	100
HCL	101
HF	110

4.2 Objective Function and Fitting Function

The objective function gives us an indicator of how people perform in the problem space. For example, in a case where the goal is to minimize, the most appropriate person is the one whose objective function has the least value. This raw information is usually used as an intermediate step to determine the relative efficiency of individuals in the genetic algorithm. The next function in this step is the fitting function. This function is used to convert the values of the objective function into analogies for the relative compatibility and efficiency of individuals. Fit values are considered positive. Therefore, in cases where the value of the objective function becomes negative, one of the functions of the fitting function is to make these values positive.

In some cases, the value of the fit function is proportional to the number of offspring from which the chromosome is expected to be produced. A common method is to fit each individual equal to the product of the objective function's sum of the total objective functions. This method is problematic when the objective functions are negative. To solve this problem, a linear conversion and positivity are used.

Using linear conversion may cause premature convergence. The selection algorithm described

later selects chromosomes based on their fit. Using linear conversion makes the number of offspring expected from each chromosome commensurate with its efficiency. If no restriction is taken, then the chromosomes that are highly efficient will dominate the other chromosomes and the new generation will be their offspring. This increases the probability that we will reach the relative optimal points and not the general optimal ones. Similarly, if the population dispersion is low, the linear conversion will increase the time required to reach the overall optimum.

By limiting the number of chromosomes, no chromosome can produce too many children. This prevents premature convergence. It is possible to choose the fit of individuals according to their position in the population and not the values of their objective function. A variable called SP is determined to assign the tendency to choose the best person, and the fit of other people is determined by Eq. (1).

$$F(x_i) = 2 - sp + 2(sp - 1)\frac{i - 1}{N_{pop} - 1} \qquad (1)$$

F is the amount of fit, N_{pop} is the number of people and x_i is the *i*th generation of the population. For the best person i = 1 and for the worst person i = Npop. For example, if the population has 40 members and sp = 0.9, then the fit of the people is in the range [0.9, 1]. The weakest chromosome will have a fit of 0.9 and the best chromosome will have a fit of 1.1. The distance between the fitting values of two adjacent chromosomes will be 0.005.

There is another way to determine the fit of individuals according to their position in the population. This fitting method is calculated from the following formula:

$$F(x_1) = \frac{2(N_{pvp} - i - 1)}{N_{pvp}(N_{pvp} + 1)} \qquad (2)$$

In this method, the distance between the fitting values is not the same, but the sum of the fitting values becomes one.

4.3 Selection and Various Techniques

Selection is the process of determining how many times a particular individual can participate in the reproductive stage. In other words, the number of children that will be born from one person. Are determined at this stage. The selection of individuals generally involves two separate processes.

1. Determine the number of times a person is expected to participate in the reproductive stage.
2. Convert the numbers obtained from the previous step to integers.

The first stage is related to the conversion of individuals to the probability of individuals participating in the crossover stage and is mostly done in the stage of determining the fit. The second stage is the probabilistic selection of individuals, which is based on relative fit and is also called sampling.

Many selection techniques use a rotating wheel namely "Roulette Wheel". A distance from zero to the sum of the fits is considered, then their fitting values are placed side by side on this distance. The size of the distance for each person is proportional to its fit. A circle can be used instead of a straight line. The circumference of the circle is equal to the sum of the fits of all the population. This process is repeated until the required number of people is selected (Fig. 12). The rotation selection method is a random sampling with an alternative. The size of each part and the probability of selection of each individual remain constant during the selection process. Anyone whose choice is not zero can theoretically be the parent of the entire next generation.

The size of each section can be changed after selection. Each time a person is selected, the unit size decreases by one unit. If the section size is negative, then zero is considered.

The Roulette wheel is used also in another way. In this method, a copy of each part is

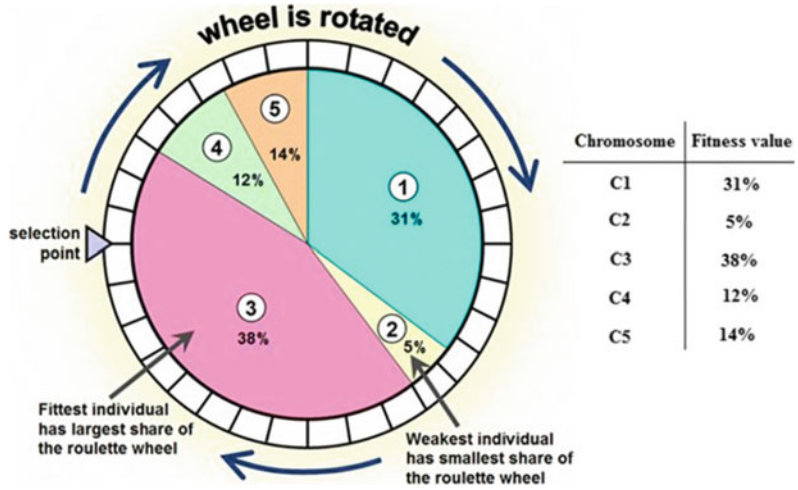


Fig. 12 Roulette wheel selection method (redrawn after Baker 1987)

selected with the correct value of the fit of each person, then the decimal values of the fits are placed in a Roulette wheel and the rest of the selections are determined in one of the previous two methods with or without replacement. A sample of pseudo code for ROULETTE WHEEL selection is presented in Fig. 13.

Another method is general random sampling. In this method, instead of selecting one person at each stage, we select all the necessary people together. If N selection is necessary, we randomly select a number such as P in the distance from zero to the result of dividing the sum of fits by N, then arrange the people in a random order. Now place the N marker in positions P, P + 1, P + N − 1. Each person on which the cursor is placed is selected. In this method less time is spent on selection.

In some articles, contest selection has been used. In this method, each time two or more chromosomes are selected by a spinning wheel and the best of them is selected and this process continues until all the necessary people are selected.

In the reference, a method was proposed to apply the restriction without applying the penalty function, which is done in the selection stage. Each time two people are selected by spinning the Roulette wheel. And a comparison is made between them. If both were allowed, the one which has higher fit is selected. If both were unauthorized, the one with less deviation to the

Fig. 13 Pseudo code of Roulette wheel selection

Function *index = Roulette Wheel Selection (**FV**)*

```
Begin
    FV = [fv_1, fv_2, ..., fv_N]^T is the unsorted fitness vector;
    // bold font indicates vector form
    [sFV, id] = sort(FV, 'ascend');      // sFV is the sorted fitness vector
                                         // id is the index of sorted individuals
    Let p = [p_i]^T, i = 1, 2, ..., N    // probability vector
    ∀i, p_i = fv_i / Σ_{i=1}^{N} fv_i;
    p = (1 − p)./Σp;      /* to assign high probability to individuals
    with low numerical value as fitness for minimization problem */
    r = rand(0,1);                       // random number between 0 to 1
    i = 1;
    While r > 0
        r = r − p_i; i++;
    End While
    index ← id(i − 1);
End
```

constraint(s) is selected. If one is permissible and the other is unauthorized, the one that is permissible is selected. In this method, there is no need for a penalty and it is independent of the type of issue, this is accounted as an advantage compared to other selection methods explained before.

In all of the above methods, the order created between the individuals should be changed before moving on to the next step and applying the operators.

4.4 Different Techniques of Cross Over

The main operator of the creation of a new generation is the stage of cross-over. Like chromosomes in nature, the off springs of this action each have some part of the information on the parent chromosomes. The simplest type of cross over is a single point cross over.

Another type of cross over operator is a multi-point cross over. In this method, a number of points on the chromosome are selected and divided into several parts. Then the same parts of the chromosomes are exchanged with each other. The first part of each chromosome is kept unchanged. This method is shown in Fig. 14. The breaking points of chromosomes are marked with a dot on them.

With the help of multi-point cross over, the problem is better explored and this improves the algorithm. At a single-point and multi-point cross-over, a number of points are defined at which the chromosome is broken and its information is exchanged with another chromosome. In a uniform cross-over, this state is extended. A random sequence of digits of a zero along chromosome length is created. Now consider the similar cells of these three chromosomes. If the number in the new random string is one, there is no change. If the number of random strings is zero, then we swap the digits of the cells of the two chromosomes.

In chromosomes that use real numbers for coding, other methods are used for intersection. One of these methods is the multiplication method. In this method, the value of the child variable is a linear combination of the parent variables.

$$x_1^{new} = \lambda_1 x_1 + \lambda_2 x_2 \tag{3}$$

$$x_2^{new} = \lambda_1 x_2 + \lambda_2 x_1 \tag{4}$$

In which x_1, x_2 are the parent variables, x_1, x_2 are the child variables, and λ_1 and λ_2 are the linear coefficients satisfies.

$$\lambda_1, \lambda_2 \geq 0 \quad \lambda_1 + \lambda_2 = 1 \tag{5}$$

It should be noted that λ_1 and λ_2 are not necessarily the same for all chromosome variables, if they are the same, it's called linear multiplication. For variables that require an integer, the result is rounded to the nearest number.

Another method of cross over is based on orientation. In this method, if the chromosome x_2 is better than the chromosome x_1 in terms of fit, then r is a random number between zero and one.

$$x^{new} = r(x_2 + x_1) + x_2 \tag{6}$$

4.5 Different Techniques of Mutation

In nature Mutation is a process in which part of a gene changes randomly. In the genetic algorithm, the probability of mutation in chromosomes is considered to be around 0.01–0.001. With the help of this operator, we can hope that good chromosomes that Removed in the selection or duplication steps. This operator also ensures that the probability of searching anywhere in the problem space is never zero, regardless of the dispersion of the initial population.

How to mutate on a binary chromosome was described in the previous sections. If we do not use binary chromosome, the mutation assigns new values to the variables within their allowable range. The more complex the chromosome encoding method, the more effective the mutation will be.

Problem space search levels will be higher than binary chromosomes by means of the

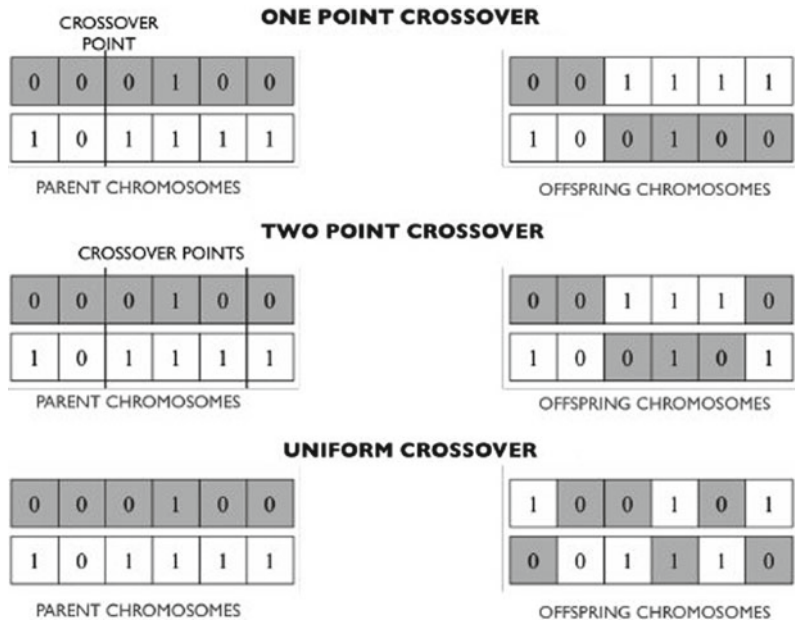


Fig. 14 Illustration of examples of one point, two points, and uniform crossover methods (adapted from source: https://www.researchgate.net/publication/268525551_Genetic_Algorithms_in_Wireless_Networking_Techniques_Applications_and_Issues/figures?lo=1)

mutation operator on chromosomes encoded in real value, and we will get better answers.

The mutation can be adjusted so that the mutation rate decreases with increasing population convergence. Real coding can be converged by limiting the mutation to small variations of the multiplication operations to achieve the answer. The following equation is proposed for mutations in real numbers:

$$x_k^{new} = x_k + \Delta\left(t, x_k^{min} - x_k\right) \tag{7}$$

$$x_k^{new} = x_k - \Delta\left(t, x_k - x_k^{min}\right) \tag{8}$$

where x_k^{max} is the largest allowable value of x_k, x_k^{min} is the smallest allowable of it, "t" is the number of generations produced so far. Also $\Delta(t_1, y)$ has a value between zero and one and is obtained from the following relation:

$$\Delta(t, y) = yr(1 - \frac{1}{T})^2 \tag{9}$$

T is the maximum number of generations. 'b' is also a parameter larger than one that determines the amount of non-uniformity. The amount of mutation decreases as the number of generations produced increases. The choice of one of the two mutant children above is random. The probability of mutation can be considered constant or it can be proportional to the number of generations produced, e.g.:

$$P_m = P_{0m} - K_{pm}\frac{t}{T} \tag{10}$$

P_m is the probability of mutation in each generation, P_{0m} is a constant value that determines the maximum probability of mutation and a constant coefficient that determines the intensity of reduction of the probability of mutation. This makes the mutation of the genes more likely to be applied first and meanwhile expands the search space. Then the probability of mutation is reduced and the search is focused on parts of the results found and the rate of convergence is increased.

5 Important Definitions in Running Genetic Algorithm

When we solve a problem, we usually seek to find the best solution, in other words, an optimal solution from among the possible solutions to the problem. The range, where the answers to the problem are acceptable so that the optimal answer is one of the subsets of this range, is

called "search space". Each point in the search space represents one of the probable solutions to the problem. Each solution is specified as the output number from which the solution is obtained, this feature and the solution related to it is called fitness.

By genetic algorithm or any of metaheuristic algorithms, we are looking for the optimal solution among all possible solutions and are specified in the search space. In fact, the search for a solution to a problem is equivalent to searching for the extreme (maximum or minimum) values of the problem within the search space. For example, an object might be to minimize the error indexes like Sum Square Error as in Least-Squared minimization. An important issue is how to prevent falling into local minimums while we are trying to find the global minimum (Fig. 15).

As shown in Fig. 15, the points inside the search space are very large and the search in this area seems so difficult and complex that we do not know from which part of this area we should start or in which area of the search space sought the optimal solution to the problem.

There are different methods for finding suitable solutions, which are mentioned below. It is noteworthy that among these methods, the genetic algorithm method offers a better solution than the others:

- Hill climbing
- Simulated annealing
- Tabu search
- Genetic algorithms.

Another application of the genetic algorithm, apart from optimization, is its use in solving difficult problems, which are known as NP-Hard Problems.

Hard problems are problems that can't be solved by common and known methods. In fact, there is no specific method and algorithm for such problems. There are other problems that are very difficult to find a solution to, but they can be solved. These types of problems are called NP-Hard problems. The method of solving such problems is based on the principle that first a solution to the problem Guess is made and then the accuracy of this guessing solution is checked so that if we use a computer to present the guesses, we can reach the answer to the problem in a reasonable time.

In the genetic algorithm method, all possible solutions to solve a problem are called chromosome, a set of these chromosomes (possible solutions) are called population. In fact, a population is a subset of the search space so that the solution or answer of a population (Solution) is obtained. It is used to form another new population. This is because the new population is better than the previous population and this process is repeated so much that we get the best answer. In the genetic algorithm, the above trend is called crossover of "Offsprings".

6 The General Process of Optimization and Problem Solving in Genetic Algorithms

Step 1. Start: The random production of a population that contains large numbers of chromosomes.

Step 2. Fitness: Evaluation of the function f(x) for chromosome x in the population.

Step 3. Creating a new population: Generating a new population by doing all of the following subgroups until a new population is created.

Step 3.1. Selection: Selection of parental chromosomes from the previous population according to its fitness so that the better the fitness (the greater the accuracy of the convergence answer) the greater the chance of selection.

Step 3.2. Cross over: Perform offspring's reproduction and create a new generation.

Step 3.3. Mutation: Determining the location of a child produced on a chromosome.

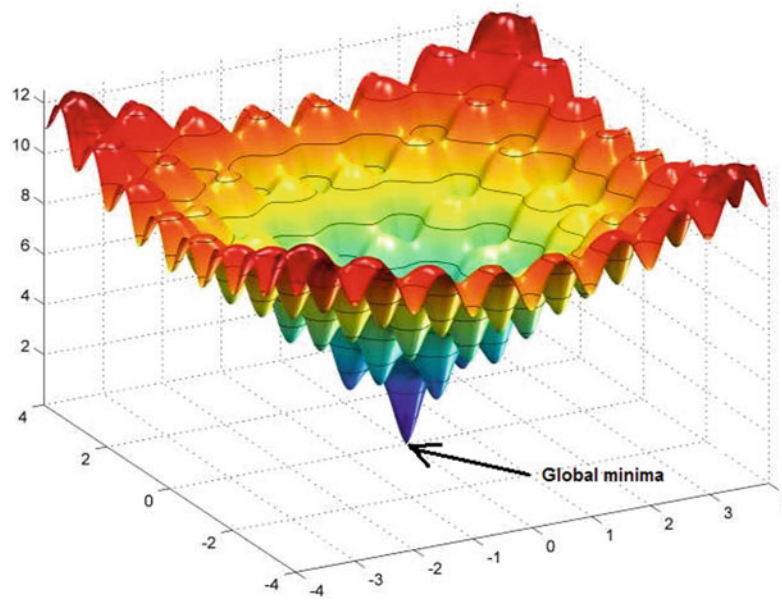


Fig. 15 An example for a complex 3D search space with a lot of local minimums, here the object of GA is to find where the global minima is located

Step 3.4. Accepting: Placing a new child inside the population.

Step 4. Replace: Replacing the new population with the previous population and using the new population in the next steps of the algorithm.

Step 5. Test: If the desired conditions are satisfied at the solution of the problem, we announce that we have reached the best answer and exit the algorithm, otherwise we go to step 2, fitness, and repeat the same process again.

The following is a schematic diagram of a genetic algorithm (Fig. 16).

7 More Examples on Operators in Genetic Algorithms

As mentioned in the previous sections, in the process of genetic algorithm, cross over and mutation are important parts of the genetic algorithm, so that the performance of this algorithm is directly affected by these two functions. Before we go into more detail about these operators, the encoding and addressing of chromosomes in a genetic algorithm is discussed.

7.1 Encoding a Chromosome

As mentioned, chromosomes contain information that is directly related to problem-solving methods. The binary method is most commonly used in chromosome encoding and addressing. As an example, two binary chromosomes, with 16 bits, are listed in Table 5.

As shown in the table above, each chromosome is made up of a string of zeros and ones. Each bit of this string identifies one of the properties used for the problem in the existing solution, and in fact the set of all these bits (the whole string) represents the solution to the problem or the corresponding chromosome that has been mentioned before. Of course, there are different methods for encoding and addressing, each of which is directly related to the type of problem and the method of solving it. Some of encoding methods are explained in Sect. 8.5, meanwhile, the binary method has more application and importance in solving problems.

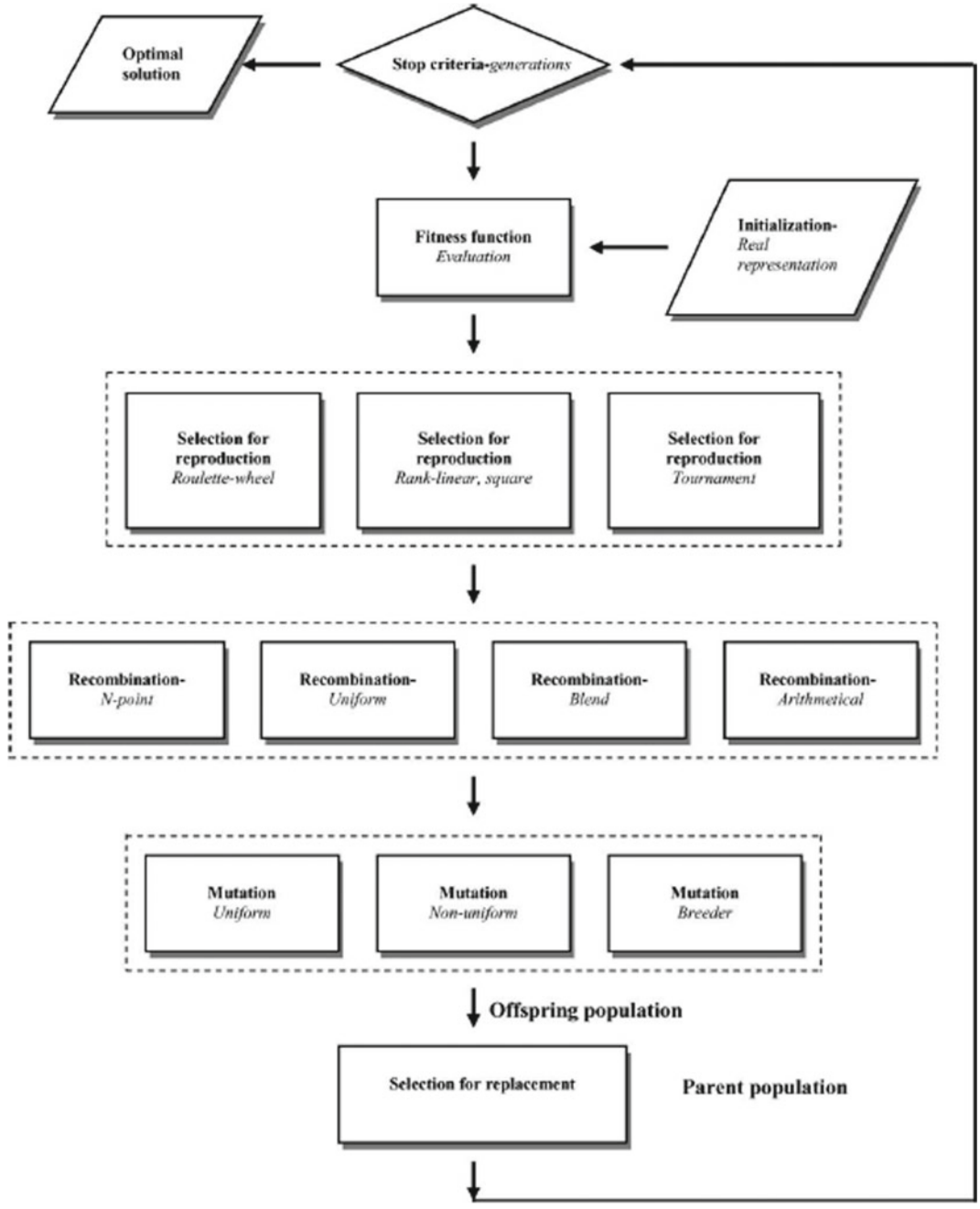


Fig. 16 schematic diagram of a general genetic algorithm

Table 5 Representation of two binary chromosomes with 16 bits

Chromosome	1101100100110110
Chromosome	1101111000011110

7.2 Cross Over

Once we have decided which method to use to encode and address the chromosomes, it is time to crossover. Crossover takes place by combining the chromosomes of the parents. The method of crossover is by first randomly selecting the point from which crossover is to begin. The order of the chromosomes of the parent's chromosomes is arranged, which is also shown in Table 6.

In the table above, chromosomes 1 and 2 are in the role of parents, and their crossover is stored in strands called offspring. Note that the symbol ¶ corresponds to the starting point of crossover and in the offspring strings the numbers after the starting point of crossover correspond to their respective chromosomes. So, the numbers after the starting point for offspring1 are for the numbers after the starting point for chromosome 1 and the numbers after the starting point for offspring are for the numbers after the starting point for chromosome 2.

There are different methods for crossover, for example we can use multiple random points instead of selecting a random point for crossover.

The last point is that choosing a suitable crossover method to solve and optimize problems in the genetic algorithm can increase the efficiency of using this algorithm many times.

7.3 Mutation

After the cross over stage, it is the turn of the mutation stage to change abruptly. The purpose of mutation is to prevent to prevent all possible solutions in a population (set of answers) being in the set of local optimal extremes. The set of optimal solutions will be determined by certain parameters and care must be taken that sub-local or local solutions are not included in the set of optimal solutions. The mutation process is applied to offspring strands completely randomly. In a binary system, the mutation is a sudden and random change from zero to one or one to zero in the offspring strings, which is also mentioned in Table 7.

The mutation technique, like the crossover technique, is directly related to the encoding and addressing of chromosomes.

Table 6 A sample of cross over producing offspring from parents

Chromosome	11011¶00100110110
Chromosome	11011¶00100110110
Offspring 1	11011¶11000011110
Offspring 2	11011¶00100110110

8 Investigation of Important Factors in Genetic Algorithm

Totally there are 6 important factors that should be investigated in any genetic algorithm, which are:

- Cross over rate
- Mutation rate
- Population size
- Selection
- Encoding of chromosomes
- Cross over and mutation methods.

These important features are explained in the following sections.

8.1 Cross Over Rate

This section refers to the quality of the cross over process. If the crossover process is eliminated in the genetic algorithm, then the offspring strings are exactly the same as the parent strings. In

Table 7 Example of binary chromosomes mutation

Original offspring 1	1101111000011110
Original offspring 2	1101100100110110
Mutated offspring 1	1100111000011110
Mutated offspring 2	1101101100110110

other words, the numbers in the offspring strings are the same as the corresponding chromosome strings.

If the cross over rate is 100% then the reproduction operation is complete and as previously mentioned, but if the crossover rate is 0%, then the offspring strands are exactly the same numbers as the corresponding chromosomes. The population produced in the later stages of the algorithm is different from the previous population, although no reproduction has taken place, which is one of the features of the genetic algorithm.

Usually the crossover rate in genetic algorithm is high and is about 80–95%, but for some problems the crossover rate of 60% is very suitable.

8.2 Mutation Rate

This section describes the quality of the mutation on chromosomes. This rate determines how many chromosomes should be mutated in one generation; mutation rate is in the range of [0, 1]. The purpose of mutation is to prevent the GA from converging to local optima, but if it occurs very often, GA is changed to random search. If the mutation rate is 0%, then the offspring strands are formed immediately after the reproduction process without any change. But if the direction is 100%, then the whole structure of the chromosome changes after production. The mutation rate is usually low, around 0.5–1%.

8.3 Population Size

Population size indicates the number of chromosomes in a population. If the number of chromosomes is large, then the rate and number of reproductions also increase. But as the number of chromosomes increases, the efficiency of the genetic algorithm decreases. In this way, the speed of the algorithm is greatly reduced and more time is spent solving and optimizing the problem.

Experience has shown that to achieve better solutions to solve the problem, large populations should be used. Although the solution time increases, but the accuracy of the answers is more important and better convergence. The appropriate size for a population is usually between 20 and 30, but for some issues, a size between 50 and 100 seems very useful. Research shows that population size also depends on the number of chromosome bits, so that if we use 32-bit chromosomes, the population size is larger than if we use 16-bit chromosomes.

8.4 Selection Methods

As mentioned in the previous sections, chromosomes are selected from a population rather than participating in regeneration as parents, but the important point is how these chromosomes are selected. There are many methods to choose from, including:

1. Roulette wheel method
2. Boltzmann method
3. Tournament method
4. Regular and back to back method namely Rank method
5. Steady state method
6. The method of Elitism, which is named after its discoverer, Mr. Elitism.

Usually, two methods of steady state and Elitism method are used in the selection process.

In the steady state method, a number of chromosomes that have higher fitness than the others are selected to create a new offspring that replaces the previous offspring.

The Elitism method is similar to the steady state method, but with the difference that in this method all the solutions that may be one of the optimal solutions to the problem are preserved and not eliminated. This is because one of the solutions may be preferable to the other from a particular point of view, but in general, taking into account all the features, the overall advantage is less, and vice versa.

The simplest method of selection is the Roulette Wheel method. In this method, first the fitness of all chromosomes in a population is calculated in the order of naming, then the

genetic algorithm randomly selects a number. The selected (random number mentioned above) is selected by the genetic algorithm. The first chromosome that has this property is selected. Figure 17 further explains the process of this selection method. Here, we have five chromosomes called individual1 to individual5, where the fitness values for each chromosome are listed in the table on the left of Fig. 17. The genetic algorithm generates the number 21 randomly. As "21" is more closed to "24" among other individuals (see the percentages shown in the pie chart), individual 2 is selected (Fig. 17).

8.5 Different Types of Encoding

Chromosome encoding and addressing is the first question that arises in the mind of a person who intends to solve and optimize problems with a genetic algorithm approach. In fact, the encoding and addressing of chromosomes depends on the

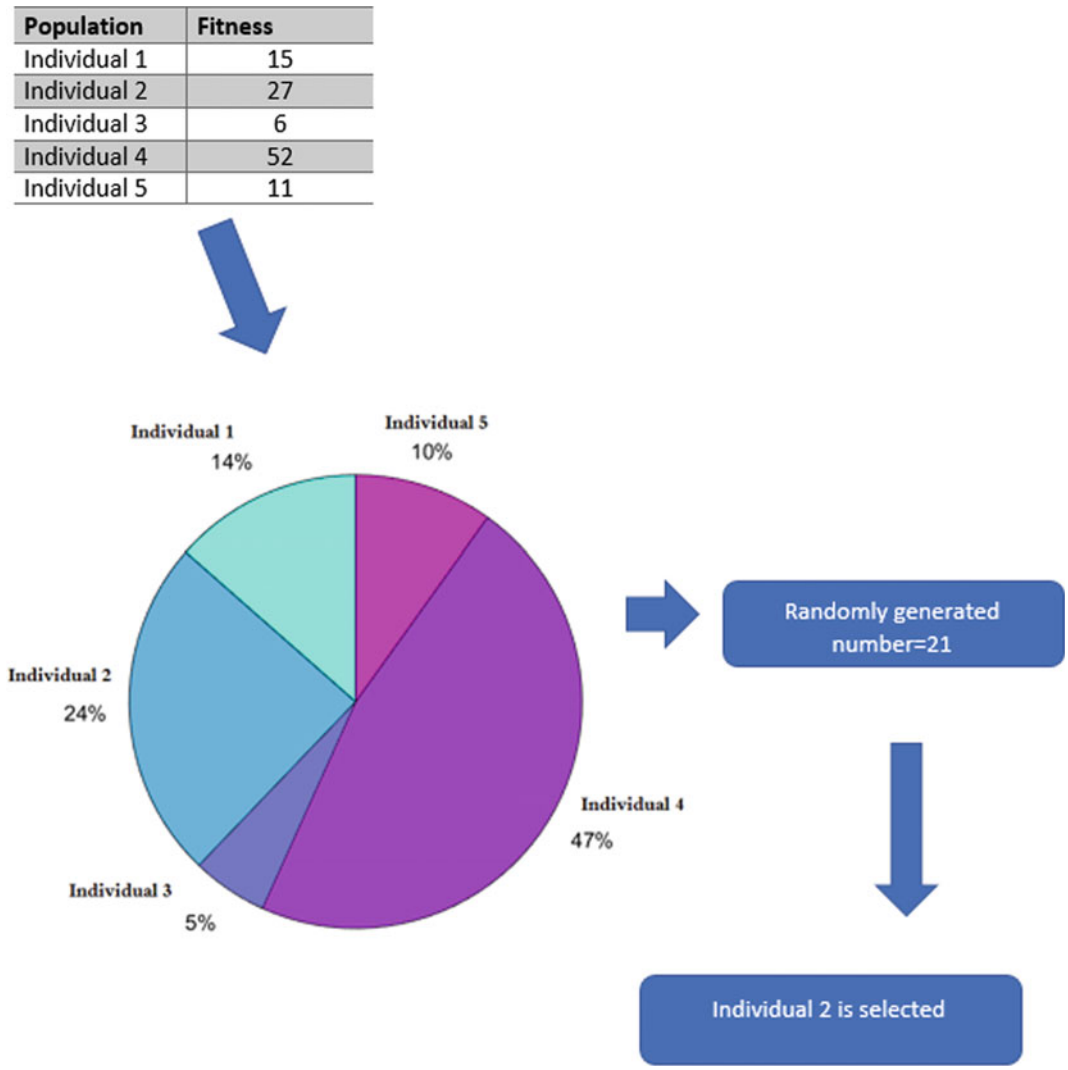


Fig. 17 An example of detailed process of selection based on Roulette wheel technique

type of problem and its application. This section refers to some of the methods of encoding and addressing chromosomes that are currently used in genetic algorithms, as we described binary encoding in the previous sections, some other methods are explained here.

8.5.1 Permutation Encoding

This method is used in sequential problems, in other words, in problems that the problem data has a special order in the problem, which is called ordering problems. One such issue is Traveling Sale Man, which deals with Hampton graphs. For example, in TSM problems, each bit of the chromosome string represents the distance from one city to a nearby city, as well as relative to other cities. In this method of encoding and addressing chromosomes, chromosome strings are composed of integers that each bit that represents a separate integer has a direct relationship with the type of application and its characteristics in case of problems. Table 8 shows an example of this type of encoding method.

In this method, for doing cross over one point is selected as the starting point, then the left bits of the first chromosome are copied to the offspring sequence to the starting point of crossover. The second chromosome then goes into action and starts checking the bits in the offspring string. In this case, if the number on the second chromosome is not in the offspring string, it copies that number in its corresponding permutation in the offspring string. This process continues until the offspring string bits are completed. Figure 18 shows the process mentioned above.

In Fig. 18, the starting point for crossover is from the fifth left bit of chromosomes A and B. After copying the first five bits on the left side of the chromosome in the string, the comparison operation is started by the chromosome, the above process and the said explanations are consistent.

In this method, to do the mutation two of the generated chromosomes are selected and then their positions in the string are switched (swapped) and the results are stored in the mutant string. This trend is shown in Fig. 19.

As shown in Fig. 19, the numbers 2 and 8 are selected and their corresponding permutations are replaced in the mutant string.

8.5.2 Value Encoding

This type of encoding and addressing is similar to permutation mode, but with the difference that instead of integers, strings with real bits or characters or a combination of the two are used, and this method is mostly used for the design and simulation of computer networks and neural networks. Table 9 shows an example of this method of encoding and addressing chromosomes.

The crossover operation in this method is exactly the same as the corresponding method in the binary method, except that the bits of the corresponding strings are not just zero and one this time, but the crossover operation with integer or real number values or character bits is done.

In the mutation process, two different bits are selected, then the value of a small integer is

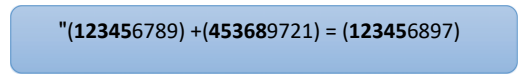


Fig. 18 The permutation process encoding

Fig. 19 Swapping operation in permutation encoding

Table 8 An example of permutation encoding

Chromosome A	1	5	3	2	6	4	7	9	8
Chromosome B	8	5	6	7	2	3	1	4	9

Table 9 An example of value encoding

Chromosome A	1.2324	1.3243	0.4556	2.3293	2.4545
Chromosome B	ABDIEIEJDHDIERDLDFLFEGT				
Chromosome C	(back)	(back)	(right)	(forward)	(left)

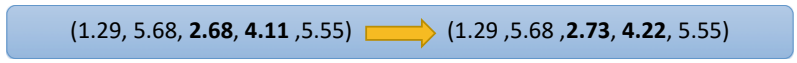


Fig. 20 Mutation process in value encoding

added or subtracted from them. This process is shown in Fig. 20.

As shown in Fig. 20, two bits are selected and a real value is subtracted from them. Note that the value deducted does not necessarily have to be the same for both bits (as in the above process where the values deducted from the selected bits are different). The same operation as above is used for character strings, except that first two bits of the generated chromosome are selected and then the corresponding character string is transformed (for example, converting 'a' to 'f' or another character).

8.5.3 Tree Encoding and Addressing

This method is mostly used for genetic programming (GP) which is done by LISP software. In this method, each chromosome contains a tree, each bit of which consists of functions and commands related to the above programming language. Figure 21 shows an example of this method. This is mostly used in the software compiler of the above software, but also in software that is often written as "genetic algorithm codes" for a specific problem, this type of method of encoding and addressing chromosomes is used.

In the crossover process, first a point of crossover is selected from both chromosomes, then the chromosomes are divided into two parts (from the crossover location), after that the upper part of the selected part is placed in the offspring string without any change, and the lower string is changed until producing a new child. These changes are shown in Fig. 22.

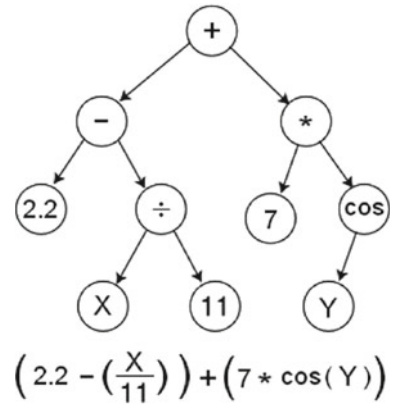


Fig. 21 An example of tree encoding (reference: https://en.wikipedia.org/wiki/Symbolic_regression)

The mutation process also uses the change of operators (for example, the addition to subtraction or the division to multiplication) and sometimes the change of numbers (Fig. 23).

9 Genetic Algorithm in MATLAB; A Brief View

An easy way to run genetic algorithms in MATLAB is using 'ga' command.

```
``ga'' attempts to solve problems of the
following forms:
    min F(X) subject to:
A*X ≤ B,
Aeq*X = Beq (linear constraints)
```

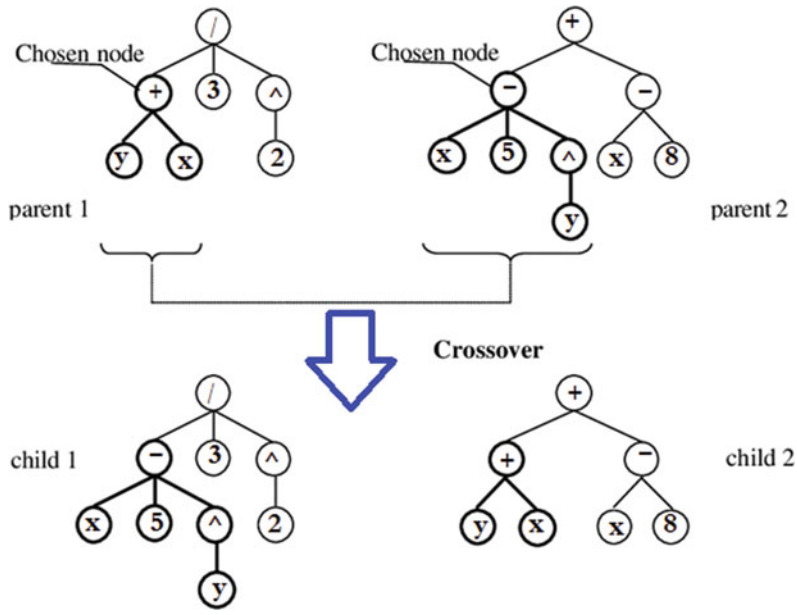


Fig. 22 An example of crossover in tree encoding

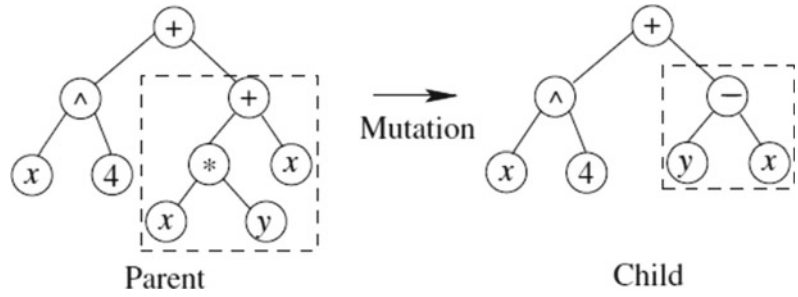


Fig. 23 An example of mutation in tree encoding (reference: https://arxiv.org/pdf/1903.01893.pdf)

$C(X) \leq 0$, $C_{eq}(X) = 0$ (nonlinear constraints)
$LB \leq X \leq UB$
X(i) integer, where i is in the index vector INTCON (integer constraints)
Note: If INTCON is not empty, then no equality constraints are allowed.
That is:
* Aeq and Beq must be empty
* Ceq returned from NONLCON must be empty

Different commands of "ga" in Matlab are listed in Table 10.

Briefly the command "***X = ga(PROBLEM)***" finds the minimum for PROBLEM. PROBLEM is a structure that has the following fields:

fitnessfcn: <Fitness function>
nvars: <Number of design variable>
Aineq: <A matrix for inequality constraints>
bineq: <b vector for inequality constraints>
Aeq: <Aeq matrix for equality constraints>
beq: <beq vector for equality constraint>
lb: <Lower bound on X>
ub: <Upper bound on X>
nonlcon: <Nonlinear constraint function>
intcon: <Index vector for integer variables>
options: <Options created with optimizations>
rngstate: <State of the random number generator>

Table 10 Different commands of "ga" in Matlab

Type of ga command	Description
X = ga(FITNESSFCN,NVARS)	Finds a local unconstrained minimum X to the FITNESSFCN using ga. NVARS is the dimension (number of design variables) of the FITNESSFCN. FITNESSFCN accepts a vector X of size 1-by-NVARS, and returns a scalar evaluated at X
X = ga(FITNESSFCN,NVARS,A,b)	Finds a local minimum X to the function FITNESSFCN, subject to the linear inequalities A * X $\leq$ B. Linear constraints are not satisfied when the PopulationType option is set to 'bitString' or 'custom'
X = ga(FITNESSFCN,NVARS,A,b,Aeq, beq)	Finds a local minimum X to the function FITNESSFCN, subject to the linear equalities Aeq * X = beq as well as A * X $\leq$ B (set A = [] and B = [] if no inequalities exist). Linear constraints are not satisfied when the PopulationType option is set to 'bitString' or 'custom'
X = ga(FITNESSFCN,NVARS,A,b,Aeq, beq,lb,ub)	Defines a set of lower and upper bounds on the design variables, X, so that a solution is found in the range lb $\leq$ X $\leq$ ub. Use empty matrices for lb and ub if no bounds exist. Set lb(i) = −Inf if X(i) is unbounded below; set ub(i) = Inf if X(i) is unbounded above. Linear constraints are not satisfied when the PopulationType option is set to 'bitString' or 'custom'
X = ga(FITNESSFCN,NVARS,A,b,Aeq, beq,lb,ub,NONLCON)	Subjects the minimization to the constraints defined in NONLCON. The function NONLCON accepts X and returns the vectors C and Ceq, representing the nonlinear inequalities and equalities respectively. ga minimizes FITNESSFCN such that C(X) $\leq$ 0 and Ceq(X) = 0. (Set lb = [] and/or ub = [] if no bounds exist) Nonlinear constraints are not satisfied when the PopulationType option is set to 'bitString' or 'custom'
X = ga(FITNESSFCN,NVARS,A,b,Aeq, beq,lb,ub,NONLCON,options)	Minimizes with the default optimization parameters replaced by values in OPTIONS OPTIONS can be created with the OPTIMOPTIONS function. See OPTIMOPTIONS for details. For a list of options accepted by ga refer to the documentation
X = ga(FITNESSFCN,NVARS,A,b,[],[],lb, ub,NONLCON,INTCON)	requires that the variables listed in INTCON take integer values. Note that ga does not solve problems with integer and equality constraints. Pass empty matrices for the Aeq and beq inputs if INTCON is not empty
X = ga(FITNESSFCN,NVARS,A,b,[],[],lb, ub,NONLCON,INTCON,options)	minimizes with integer constraints and the default optimization parameters replaced by values in OPTIONS. OPTIONS can be created with the OPTIMOPTIONS function. See OPTIMOPTIONS for details

10 Random Numbers Generation in Matlab

As in the most parts of the GA, producing random numbers is required in this section we introduce some useful commands and functions in Matlab about it.

10.1 Random Permutation

P = randperm(N) returns a vector containing a random permutation of the integers 1:N. For example, randperm(6) might be [2 4 5 6 1 3]. P = randperm(N,K) returns a row vector containing K unique integers selected randomly from 1:N. For example, randperm(6,3) might be [4 2 5].

randperm(N,K) returns a vector of K unique values. This is sometimes referred to as a K-permutation of 1:N or as sampling without replacement.
To allow repeated values in the selection, sometimes referred to as sampling with replacement, use RANDI(N,1,K)
"randperm" calls RAND and therefore changes the state of the random number generator that underlies RAND, RANDI, and RANDN. Control that shared generator using RNG.

10.2 Pseudorandom Integers from a Uniform Discrete Distribution

In this way command "randi " is introduced in MATLAB.

R = randi(IMAX,N) returns an N-by-N matrix containing pseudorandom integer values drawn from the discrete uniform distribution on 1:IMAX.

Table 11. shows different types of this command.

Table 11 List of different types of "randi" command in Matlab with examples

Command	Description	Example in Matlab*
randi(IMAX,M,N) or randi(IMAX,[M,N])	Returns an M-by-N matrix	r =randi$(9,2,2)$ r =4 4 5 6
randi(IMAX,M,N,P,…) or randi(IMAX,[M,N,P,…])	Returns an M-by-N-by-P-by-array	w = randi$(9,1,2,3)$ w$(:,:,1) = 4$ 9 w$(:,:,2) = 9$ 1 w$(::,3) = 7$ 3
randi(IMAX)	Returns a scalar	>> randi(14) ans = 8 >> randi(14) ans = 14
randi(IMAX,SIZE(A))	Returns an array the same size as A	>> A = [1 4 sqrt(2)*5; −7 78 6.9]; >> w = randi(10,size(A)) w = 5 4 7 10 8 6
randi([IMIN,IMAX],…)	Returns an array containing integer values drawn from the discrete uniform distribution on IMIN:IMAX	>> randi([14 77],3) ans = 34 41 44 24 20 58 25 52 58 >> randi([14 77],2,2) ans = 54 18 16 34
randi(…,CLASSNAME)	Returns an array of integer values of class CLASSNAME	>> D = randi(10,4, 'distributed') D = 2 2 7 4 7 7 6 8 10 3 2 5 6 10 7 4

Note The size inputs M, N, P, … should be nonnegative integers
Negative integers are treated as 0, *Note that as these commands generate random numbers in anytime of running, they might generate different values

Table 12 List of different types of "randn" command in Matlab with examples

Command	Discription	Example in Mtlab
R = randn(N)	Returns an N-by-N matrix containing pseudorandom values drawn from the standard normal distribution	>> randn(3) ans = −0.0638 1.8140 −0.7231 0.6113 0.3120 0.5265 0.1093 1.8045 −0.2603
randn(M,N) or randn([M,N])	Returns an M-by-N matrix	>> randn(2, 3) ans = 0.6001 −2.1860 −1.4410 0.5939 −1.3270 0.4018
randn(M,N,P,…) or randn([M,N,P,…])	Returns an M-by-N-by-P-by-… array	>> w = randn(1, 3, 2) w(:, :, 1) = − 0.6264 0.2495 − 0.9930 w(:, :, 2) = 0.9750 − 0.6407 1.8089
randn	Returns a scalar	>> randn ans = −1.0799 >> randn ans = 0.1992
randn(SIZE(A))	Returns an array the same size as A	>> A = [1 4 5; 5 − 1 3]; >> randn(size(A)) ans = −1.5210 −0.5933 0.9421 −0.7236 0.4013 0.3005
R = randn(…, CLASSNAME)	Returns an array of normal values of the specified class. CLASSNAME can be 'double' or 'single'	>> L = randn(2,1,'double') L = −0.3731 0.8155
R = randn(…, 'like', Y)	Returns an array of normal values of the same class as Y	

10.3 Normally Distributed Pseudorandom Numbers

In this way command "randn" is introduced in MATLAB. There are various states for using this command listed in Table 12.

References

Baker JE (1987) Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the second international conference on genetic algorithms. Morgan Kaufmann Publishers, pp 14–21

Fogel GB (2012) Evolutionary programming. In: Rozenberg G, Bäck T, Kok JN (eds) Handbook of natural computing. Springer, Berlin. https://doi.org/10.1007/978-3-540-92910-9_23

Fogel LJ (1962a) Autonomous automata. Indus Res 4:14–19

Fogel LJ (1962b) Toward inductive inference automata. In: Proceedings of the congress. International Federation for Information Processing, Munich

Holland JH (1962) Outline for a logical theory of adaptive systems. J ACM 9(3):297–314. https://doi.org/10.1145/321127.321128

https://arxiv.org/pdf/1903.01893.pdf

https://commons.wikimedia.org/w/index.php?curid=83685527

https://earthquake.usgs.gov/learn/glossary/?term=dip

https://en.wikipedia.org/wiki/Symbolic_regression

https://mathworld.wolfram.com/GrayCode.html

https://micro.magnet.fsu.edu/cells/animals/animalmodel.html

https://sites.google.com/site/ldgeohazards/what-are-earthquakes/fault-systems

https://www.civilsdaily.com/biotechnology-basics-of-cell-nucleus-chromosomes-dna-genes-etc/

https://www.pngitem.com/middle/hRoTRhb_transparent-chromosome-clipart-chromosomes-and-genes-hd-png/

https://www.researchgate.net/publication/268525551_Genetic_Algorithms_in_Wireless_Networking_Techniques_Applications_and_Issues/figures?lo=1
https://www.semanticscholar.org/paper/An-Adaptive-and-Memory-Assisted-Local-Crossover-in-Iqbal-Hoque/b1ea9c73ee18c7126238dcc0bddc0725160af541/figure/11
https://www.usgs.gov/faqs/what-gases-are-emitted-kilauea-and-other-active-volcanoes
MATLAB Software (2018) MathWorks
Rechenberg I (1965) Cybernetic solution path of an experimental problem. Royal Aircraft Establishment, Library Translation 1122, Farnborough
Schwefel HP (1968) Experimentelle Optimierung einer Zweiphasendüse. Ber. 35 AEG Forsch. Inst. Proj. MHD-Staustrahlrohr (Nr.l 1034/68), Berlin
Shopova EG, Vaklieva-Bancheva NG (2006) BASIC—a genetic algorithm for engineering problems solution. Comput Chem Eng 30:1293–1309

Application of Genetic Algorithm in Volcanology and Seismology

Abstract

In this chapter after presenting a briefed list of the GA applications, various advanced applications of Genetic Algorithms in the fields of seismology and volcanology are presented. The examples are arranged so that cover both the seismological and volcanological aspects.

1 Introduction

Genetic algorithms are widely used in both volcanology and seismology in various disciplines, the high ability of GA to search even in enormous search spaces is a good motivation to use it for invers problems in geophysics. Four main aspects that GA has been used in volcanology and seismology is shown in Fig. 1. Also, some of the related researches are listed in Table 1.

Given that there is a lot of research on the use of genetic algorithms in volcanology and seismology, it is certainly beyond the scope of this book to review all of them, and we refer those interested to the references at the end of this chapter for further reading. However, in order to become more familiar with how to use the genetic algorithm in these fields, some examples of applications in volcanology and some examples in seismology are presented in more detail later in the chapter.

2 Inverse Modelling of Volcanomagnetic Fields Using Genetic Algorithm

2.1 Mechanisms that Cause Volcanomagnetic Anomalies

Currenti et al. (2005) used a genetic algorithm technique to inverse the volcanomagnetic anomalies which mostly caused by the such as thermomagnetic, piezomagnetic and electro kinetic effects. The local magnetic field variation is affected during the volcanic activity processes. From the viewpoint of volcanos system, the main reason is that they act like a heat-transfer, and consequently their activity is frequently accompanied by hydrothermal convection of the subsurface which leads to variation of pressure, temperature, and fluid motion within the edifice. The modifications within the volcanic edifice of the stress field and/or of the thermodynamic state is induced as variations in the magnetization of the rocks, which generate a wide variety of magnetic signals (Currenti et al. 2005). The most important geophysical mechanisms producing magnetic anomalies during the volcanic activities are classified into three main categories depicted in Fig. 2.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

A. Hajian et al., *Intelligent Methods with Applications in Volcanology and Seismology*, Advances in Volcanology, https://doi.org/10.1007/978-3-031-15432-4_7

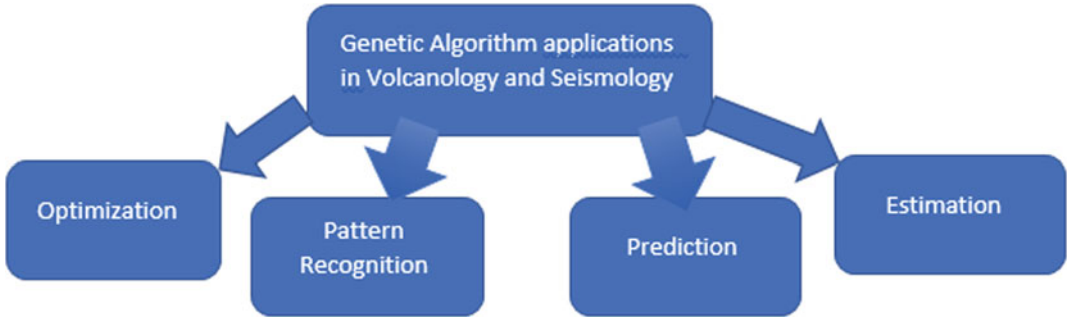


Fig. 1 Main aspects of GA applications in volcanology and seismology

Table 1 Some researches on GA applications in volcanology and seismology at a glance

Genetic application in volcanology/seismology	Researcher(s)/year of publication
Genetic algorithms for earthquake location using teleseisms	Kennett and Sambridge (1992)
Staged hybrid genetic search for seismic data imaging	Mathias et al. (1994)
Niching genetic algorithm (NGA) for inversion of teleseismic body waves for the source parameters	Koper et al. (1999)
Modeling broadband regional waveforms for crustal structure in the Western United States	Bhattacharyya et al. (1999)
Inverse modelling of volcanomagnetic fields using genetic algorithm	Currenti et al. (2005)
Automated horizon correlation across faults in seismic images	Aurnhammer and Tönnies (2005)
Inversion of SAR data in active volcanic areas by genetic algorithm	Nunnari et al. (2005)
Multiobjective genetic algorithm inversion of ground deformation and gravity changes	Carbon et al. (2008)
Focal-mechanism determination in Taiwan by genetic algorithm	Wu et al. (2008)
Seismic wavelet estimation through chaos-genetic algorithm based on the cat map	Wang et al. (2009)
Ground motion predictive modelling based on genetic algorithms	Yilmaz (2011)
Joint of the seismic attributes optimizing based on the genetic algorithm and the neural network methods	Xiong et al. (2011)
Locating seismic-sense stations through genetic algorithm	Espinosa-Ramos and Vázquez (2011)
Evaluation of generate good earthquake risk models using genetic algorithms	Aranha et al. (2014)
Localization of micro seismic source based on genetic-simplex hybrid algorithm	Li et al. (2017)
Generator of genetic seismic signals	García and Alcántara (2019)
Loss-based performance assessment and seismic network optimization for earthquake early warning	Böse et al. (2022)

2.2 Motivations to Use Genetic Algorithm for Inversion of Volcanomagnetic Anomalies

A great deal of effort is involved to model magnetic fields expected during volcanic activities especially the main eruptions i.e. paroxysm, they face three main problems listed in Table 2. Furthermore, analytical solutions of volcanomagnetic fields use magnetic models with highly non-linearity and, usually, characterized by a high number of parameters (Currenti et al. 2005). Hence, the related inverse problems involve the analysis of appropriate numerical strategy to promptly identify and interpret the source parameters of a wide spectrum of volcanomagnetic signals recorded on active volcanic areas (Del Negro and Nunnari 2003). To overcome the mentioned problems to invert the volcanomagnetic anomalies a better way is using Genetic Algorithms because they have:

1. The capability of performing a much broader search over the model parameters.
2. A greater likelihood of finding the global optimal solution.
3. Good performance to find the optimum even in presence of local minima in the objective function (Goldberg 1989).

2.3 The GA Procedure to Invert Volcanomagnetic Anomalies

Currenti et al. (2005) prepared a coding program which calculates the volcanomagnetic anomaly for any categorizing of three main geophysical

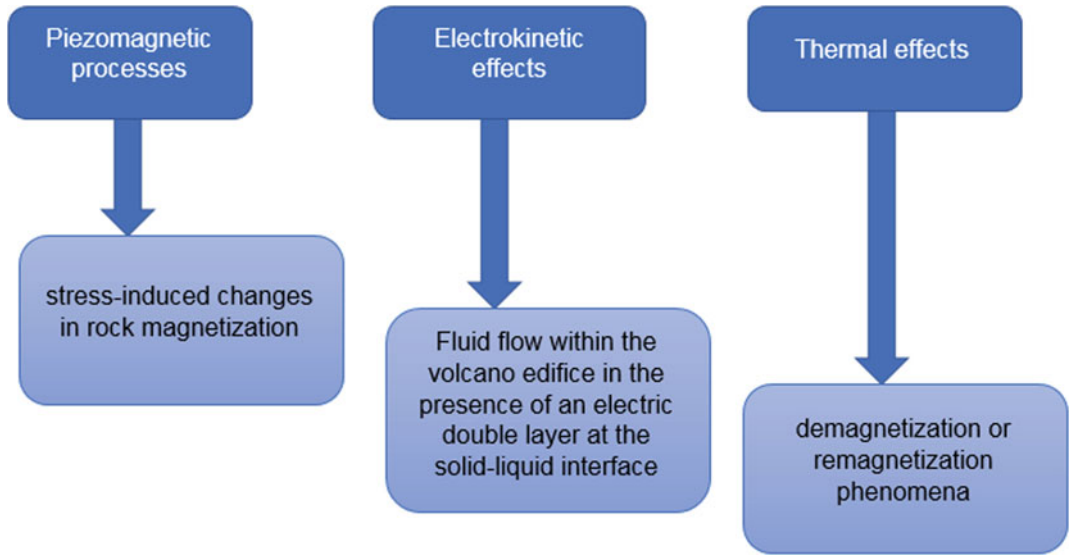


Fig. 2 Categorization of main geophysical mechanisms that cause magnetic anomalies during volcano activities

Table 2 The main problems of classical inversion methods for inversion of vocanomagnetic anomalies

Problem	Description
Scattering	The solutions are scattered throughout the literature
Limited	They may have only limited applicability to certain locations and regimes
Time consuming	Algorithms are frequently complex and iterative

mechanism that cause magnetic variations during volcanic activities namely VMM. This program inputs are the magnetic model parameters and the output is the related magnetic anomaly. In fact, when the population of magnetic source model parameters are generated the related anomaly is calculated by VMM, then this simulated anomaly is evaluated by the objective function and checked by evolutionary rules. This procedure is repeated until to reach the desired stopping criteria. The flowchart of GA used for this inversion is shown in details in Fig. 3.

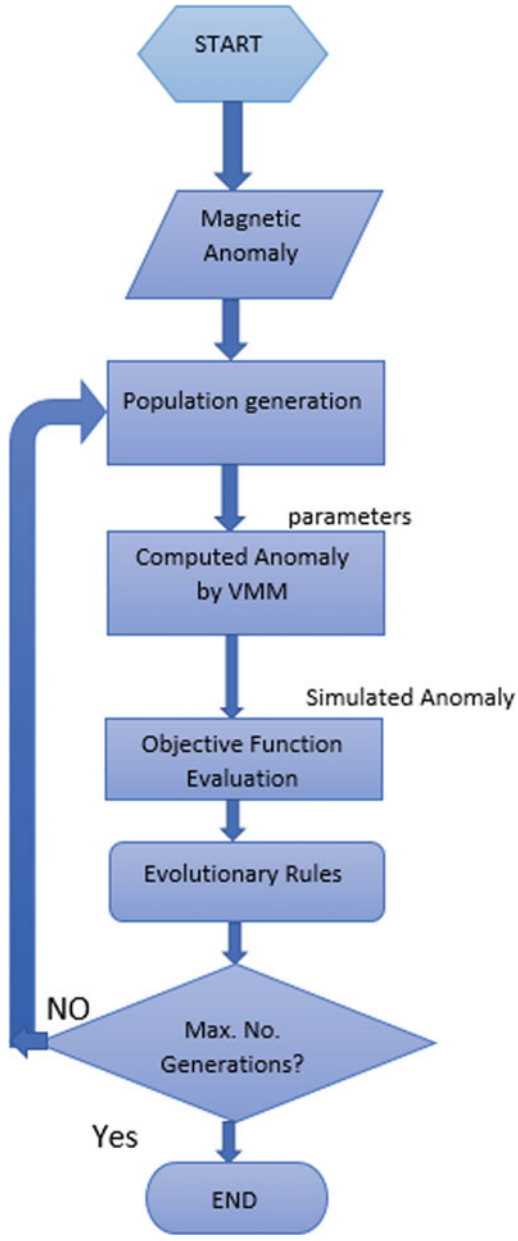


Fig. 3 The flowchart of GA method for inversion of volcanomagnetic anomaly (redrawn after Currenti et al. 2005)

2.4 Forward Models

2.4.1 Piezomagnetic Field Forward Model

The piezomagnetic field (B_{VP}) at the ground surface of a volcano can be estimated by (Zlotnicki and Le Mouel 1988):

$$B_{V_p} = -\frac{\mu_0}{4\pi}\nabla \iiint_{V_p} \frac{\Delta J(M,t)}{r^3} dV_M \qquad (1)$$

where μ_0 is the magnetic permeability in the vacuum, Q is the observation point, M is a point in the rock volume Vp submitted to a stress field, and r is the distance between M and Q. Application of this formula requires estimating the value of three parameters. The intensity of the piezomagnetic effect is proportional to the product of the stress sensitivity (β), the average magnetization (J) and the average rigidity of the Earth's crust (λ) (Currenti et al. 2005).To simulate the volcano magnetic anomaly generates by piezomagnetic process two common models namely Mogi and Okada were used, these models with related figures and descriptions are depicted in Table 3. Furthermore, the Mogi and Okada Model sources parameters with the range values of the synthetic models are shown in Tables 4 and 5, respectively.

2.4.2 Electrokinetic Effects Forward Model

The electrokinetic magnetic field (B_{VE}) observed at any time t, on the ground surface is given by (Fitterman 1978, 1981):

$$B_{v_E}(Q,t) = \frac{\mu_0}{4\pi}\nabla \times \iiint_{v_p} \frac{j(M,t)}{r} dV_M \qquad (2)$$

Table 3 Two forward models of Piezomagnetic processes

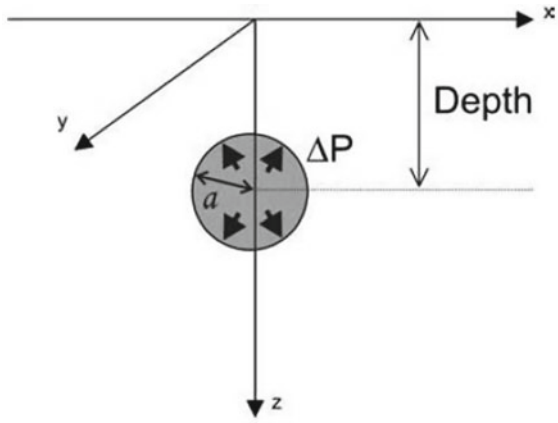
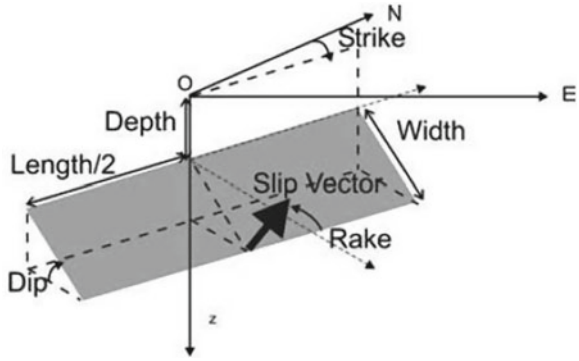
Model name	Figure of the model	Description
Mogi's source		A hydrostatically pumped pressure source in a homogeneous and isotropic elastic half-space with a uniformly magnetized upper layer centred at (0, 0, *D*) with radius *a*. The Curie point isotherm is at a depth *H* (Currenti et al. 2005)
The Okada source		A fault occurring in homogeneous and isotropic elastic half-space with a uniformly magnetized upper layer. H is the depth of the Curie point isotherm. Depth is the distance between the origin O and the upper edge of the fault; Strike is the orientation of the fault with respect to the North; Dip is angle of the fault plane with respect the horizontal plane; Rake is the angle of the displacement; Length and Width are the length and width of the fault respectively; Slip is the module of the displacement (Currenti et al. 2005)

Table 4 Mogi model sources parameters with the range values of the synthetic models (Currenti et al. 2005)

Mogi model geometrical parameters	Symbol	Unit of measurement	Ranges
Volume of the spherical source	V	m^3	10^6–10^8
Depth of the center of the sphere from the surface	D	m	200–3000
Sphere centre coordinates	[X_C, Y_C]	m	−4000–4000

with Q the observation point and *j(M, t)* the electrokinetic current at the time t (calculated by Eq. (3) and in a point M of the volume VE affected by the fluid flow (Currenti et al. 2005).

$$\mathrm{j} = -\emptyset\rho\left[\nabla E - \frac{\varepsilon\xi}{\eta\rho}\nabla P\right] = -\emptyset\rho[\nabla E + C\nabla P] \tag{3}$$

Table 5 Okada model parameters with the range values of the synthetic models (Currenti et al. 2005)

Okada model geometrical parameters	Symbol	Unit of measurement	Ranges
Dip fault	δ	Degrees	0–90
Azimuth	α	Degrees	0–180
Depth	D	m	200–3000
Fault width	W	m	200–3000
Fault length	L	m	500–6000
Dislocation	u	m	1–3
Fault centre coordinates	[X_C, Y_C]	m	−4000–4000

where the C coefficient is the streaming potential coefficient, E is the streaming potential, P is the fluid pressure, φ is the porosity of medium, ε is the dielectric constant, ζ is the zeta potential, η is the viscosity of fluid and ρ is the electrical conductivity. The electrokinetic source intensity for the case of inclined fault separating two media with different streaming potential, C_1 and C_2 is constant and bounded by the fault geometry given by:

$$S = (C_1 - C_2).\ P \tag{4}$$

The corresponding magnetic parameters that Currenti et al. (2005) used for their electrokinetic effects forward modeling are listed in Table 6.

2.4.3 Thermomagnetic Phenomena Forward Model

Currenti et al. (2005) noted that for a uniformly demagnetized sphere, the anomalous field due to the demagnetization process is equivalent to a dipole located at the centre of the sphere with the orientation anti-paralleled to the direction of the original magnetization. A spherical magnetic

Table 6 The magnetic properties are fixed to values that resembles cases likely to occur in volcanic areas (Currenti et al. 2005)

Value	Units of measurement	Symbol	Magnetic parameters
5	A m^{-1}	J	Average magnetization
15,000	m	H	Curie depth
53	Degree	I	Magnetic inclination
1	Degree	D	Magnetic declination
Piezomagnetic model parameter			
3×10^5	bar	μ	Rigidity
0.25	$0.25(\lambda = \mu)$	υ	Poisson's ratio
0.0001	bar^{-1}	β	Stress sensitivity
Electrokinetic model parameter			
20	mV/bar	C_1	Streaming potential of upper side
10	mV/bar	C_2	Streaming potential of lower side
0.01	S/m	ρ_1	Electric conductivity of upper side
0.1	S/m	ρ_2	Electric conductivity of lower side
1	V	S	Source function

source located at (0, 0, z) causes the change in the total intensity (ΔF) at point (x, y, 0) which is given by:

$$\Delta F = -\frac{M}{4\pi r^3}\left\{1 - 3\left(\frac{x}{r}\cos I - \frac{z}{r}\sin I\right)^2\right\} \quad (5)$$

where M is the intensity of the magnetic moment of the source, r is the distance between the centre and the observation point, x the horizontal projection of r and I the inclination of the ambient geomagnetic field (Hamano et al. 1990 in Currenti et al. 2005). The forward model parameters and the medium constants that Currenti et al. (2005) used in this way are listed in Table 7.

2.5 Testing and Evaluating the GA Performance for Synthetic Data

Currenti et al. (2005) produced a larg number of synthetic data by using different couples of parameters for the forward modelings mentioned with details in the last section for creating the search space. As a sample, the parameters of Okada model is showm in Table 8. Then the volcanomagnetic anomalies were selected as input of the genetic algorithm to be inverted into the related parameters and the geomagnetic anomaly was calculated from these parameters to compare with the synthetic volcanogemagnetic. The index error they used to evaluate the effieciency of the method was Normal Mean Absolut Error (NMAE) clculated by Eq. (6):

$$\mathrm{NMAE}\% = \frac{100}{(N.Range)}\sum_{i=1}^{N}|P_{C_i} - P_{T_i}| \quad (6)$$

where N is the number of considered models, P_{Ti} is the generic ith parameter of the calculated model, and P_{Ci} is the corresponding true parameter.

The results of this evaluation is depicted in Fig. 4 which emplies that GA results are acceptable for inversion of volcanomagetic anomalies.

2.6 Evaluation of GA Results for Real Cases

However, there are a large number of active volcanos around the world but less of them have enough number of stations that record the time

Table 7 Parameters for prismatic body used for thermomagnetic phenomena forward modeling (Currenti et al. 2005)

Prismatic body geometrical parameters	Symbol	Ranges (in m)
Depth	D	200–3000
Width	W	200–3000
Length	L	200–6000
Thick	T	200–1000
Fault center coordinates	[X_C, Y_C]	−4000–4000

Table 8 Parameters of the electrokinetic Okada source.

Parameter	Value	Mean value	Standard deviation
Dip	60°	50.35°	15.65°
Strike	40°	39.83°	0.36°
Length (m)	2000	2004.33	64.43
Width (m)	1000	1262.46	377.53
Depth	800 m	789.82	22.38
X_C	0°	54.73 m	67.11 m
Y_C	0°	−12.47 m	20.16 m

The mean value and the standard deviation are computed over the estimated parameters obtained by 20 different runs of GA (Currenti et al. 2005)

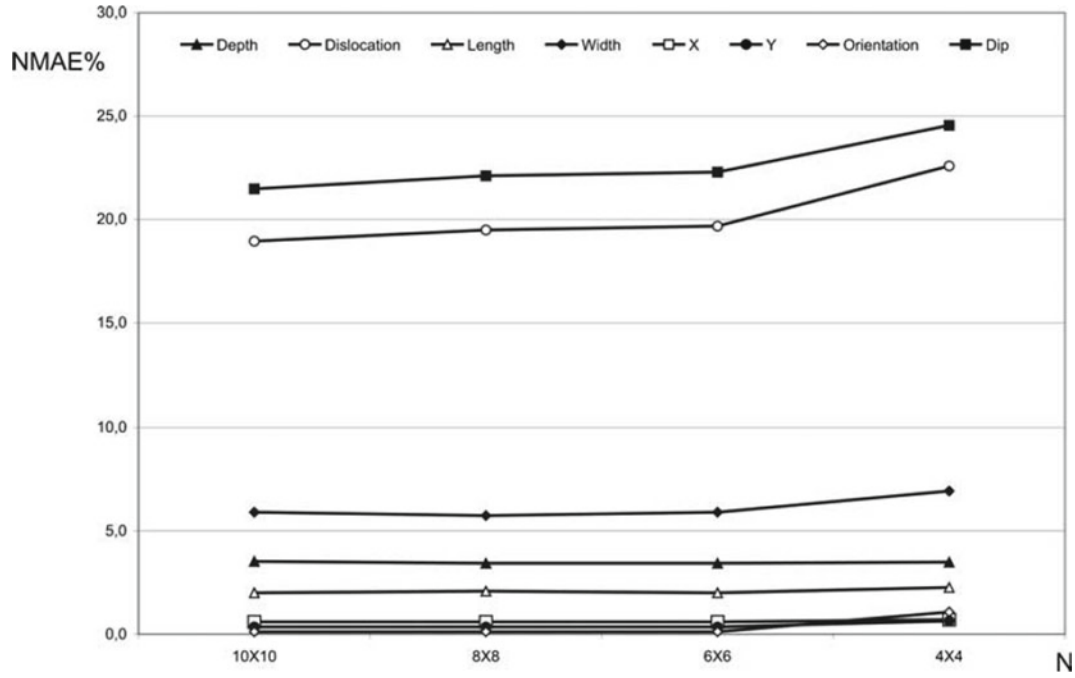


Fig. 4 The NMAE% (normal mean absolute error percentage) performance indices for the piezomagnetic Okada model (Currenti et al. 2005)

series of volcanomagnetic variations. Fortunately, Etna in Italy and Miyakejima in Japan are the ones with enough number of geomagnetic stations close to their summit. Hence Currenti et al. (2005) found two case studies to investigate the GA performance for real volcanomagnetic anomalies inversion:

(a) Thermomagnetic anomalies during an intrusive episode occurred during the 1989 Mt Etna eruption.
(b) Magnetic anomalies measured at Miyakejima volcano in terms of piezomagnetic effect.

During the 1989 eruption of Mt Etna the thermomagnetic anomalies, as shown in Fig. 5, were appeared in the recorded geomagnetic signals. Using GA the volcano magnetic anomaly was calculated and plotted (Fig. 5), the results showed very close values to that of real anomalies which is acceptable to interpret the volcanomagnetic anomalies during the episode of a volcano eruption.

Currenti et al. (2005) also tested their GA solution to invert the Magnetic anomalies during the 2000 Miyakejima volcano eruption, the results are show in Fig. 6. The outputs of both synthetic and real cases implied that GA method is suitable to face with non-linearity involved in the inversion of the volcanomagnetic anomalies.

3 Inversion of SAR Data in Active Volcanic Areas by Genetic Algorithm

3.1 Abstract

In active volcanos ground deformation is a very prevalent phenomenon reference to the dynamic of different kinds of magma sources. The efficiency of ground deformation inversion based on GA optimization has been investigated by various authors (such as Sambride and Mosegaard 2002; Tiampo et al. 2004). Nunnari et al.

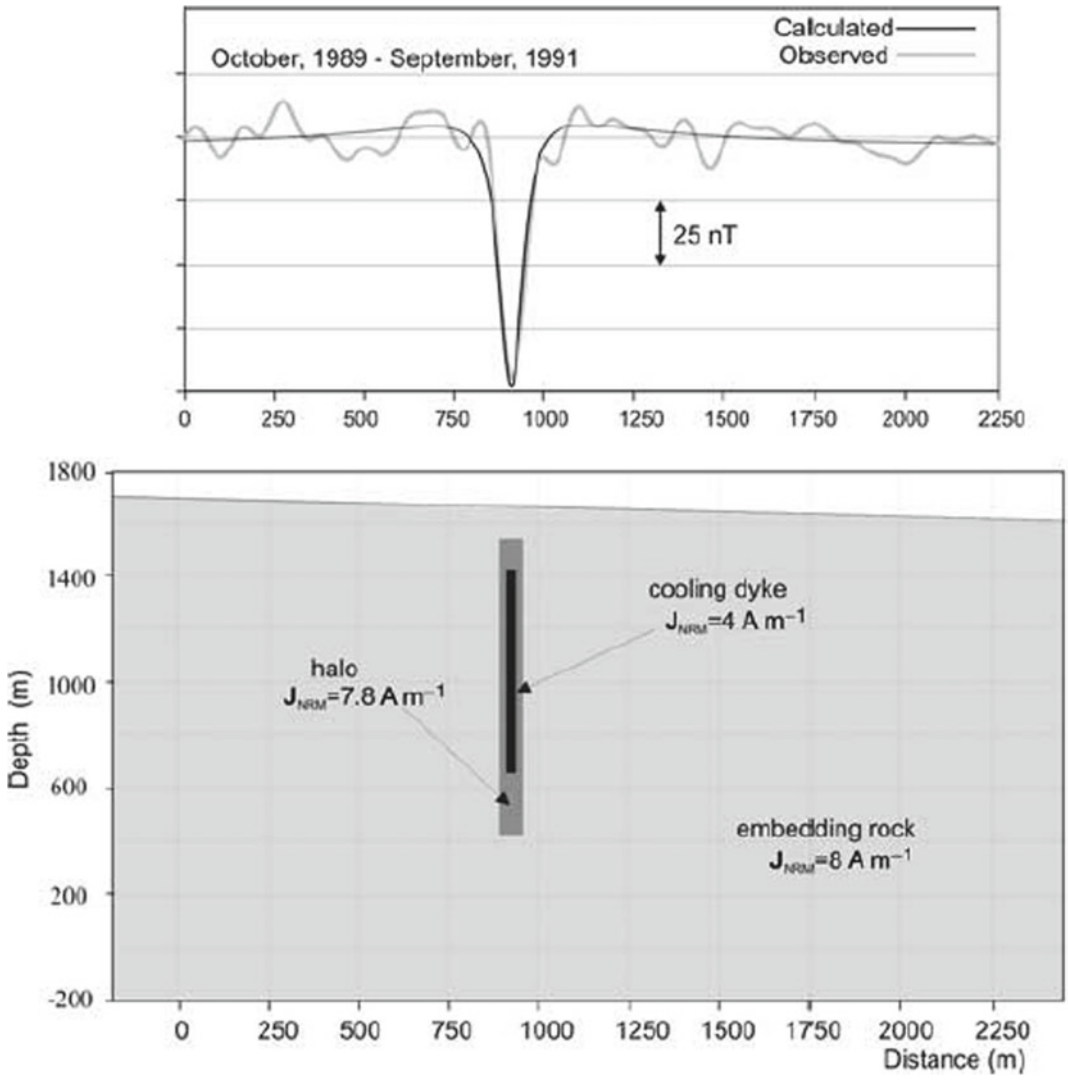


Fig. 5 Thermomagnetic anomalies at Mt Etna: observed (grey line) and computed through GA (black line) (Currenti et al. 2005)

(2005) used genetic algorithm approach for inversion of ground deformation measured through SAR (Synthetic Aperture Radar) interferometry that were recorded on Mt. Etna during eruptions occurring in 1998 and 2001. SAR interferometric data were inverted using a GA optimization algorithm. In this way, the inversion of ground deformation data was done relating to hydrostatically pumped spherical sources (Mogi) and magma-filled dikes (Okada) that are often considered to fit ground deformation data observed in active volcanic areas such as Mt. Etna. Both models are introduced in details in the next section. They assessed the goodness of the inversion procedure using both synthetic and actual SAR data.

3.2 SAR Data Forward Modelling

The traditional Mogi and Okada models for generating synthetic SAR data have been listed pictorially with related components u_x, u_y and u_z of the displacement in Table 9.

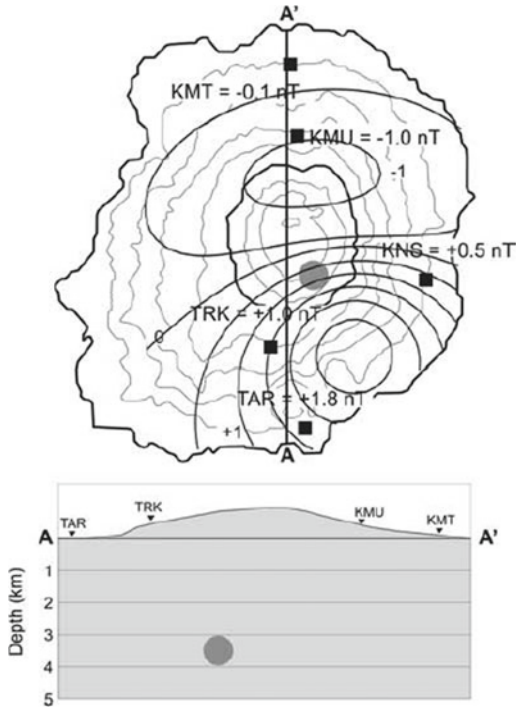


Fig. 6 The magnetic monitoring array at Miyakejima volcano (after Sasai et al. 2002). The five stations are those used for the GA inversion. The computed anomaly through GA and the amplitude of the step-like magnetic variations recorded in correspondence with the July 14 tilt-step event are also reported (Currenti et al. 2005)

3.3 Test of GA Method for Synthetic Data

To apply the GA method for inversion of SAR data (or any other types of data) one important issue is to assign a suitable cost function which is also referred as fitness function. In this way there are two common definitions listed in Table 10.

Nunnari et al. (2005) inverted hundred models uniformly distributed in the space of parameters, hypothesizing a grid with 21 × 21 vertices (i.e. 441 measuring points) and index errors of all tested model parameters were reported (Table 11) containing Bias, RMSE, NMAE% and d (index of agreement as defined in Table 10). The results showed that the GA can invert deformation caused by a single dislocation with a high degree of accuracy in the case of synthetic data.

3.4 Inversion of Real SAR Data

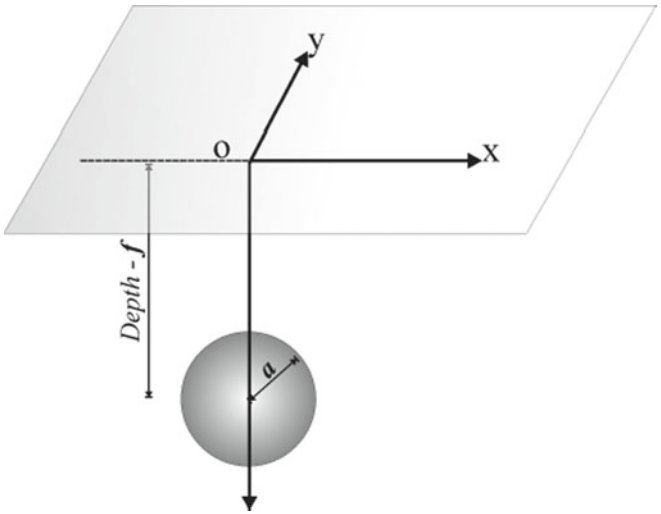
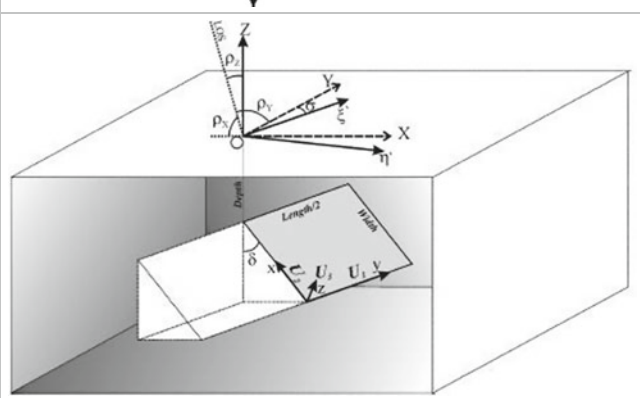
In order to evaluate the GA performance for inversion of real data Nunnari et al. (2005) selected two eruptive events of Mt. Etna during 1998 and 2001, respectively. The 1998 eruptive event was used for testing the Mogi source model and the eruption 2001 for testing the Okada model. A vigorous explosive eruption was produced by the Mt. Etna Voragine Crater on 22 July 1998. A 10 km high eruptive column above the crater rim formed at the eruptive climax between 16:48 and 17:14 GMT (Aloisi et al. 2002). A remarkable uplift was observed in the pre-event period which is followed by a short lowering phase and hence a reprise of the uplift. Nunnari et al. (2005) selected the short post event period to which the image pair 22 July–26 August 1998 refer as the object of GA inversion (Fig. 7).AS it can be seen the larger deformation effect is mostly happened at the center of the image, so this portion of the whole SAR image (shown in Fig. 8) was used to be inverted by GA. This selection simplifies the process of GA inversion and also reduces the computational effort.

Nunnari et al. (2005) applied GA inversion to find the optimal parameters of a Mogi type source. The results showed that the inversion algorithm calculates the nucleus strain (C), the source coordinates and depth with a good accuracy because they have good adaption with other available geological evidences (Table 12).

Also as another real case study to test GA with Okada model, Nunnari et al. (2005) used a descending interferogram (Fig. 9a) referring to the image pair 15 November 2000–31 October 2001 of Mt. The low quality of the SAR image suggested using a resized area (Fig. 9b) characterized by high coherence.

The results of the GA inversion for Etna (October 2001 eruption) are listed in Table 13 first column, it's necessary to mention that to evaluate the accuracy of estimated parameters, an independent inversion (using independent GPS data) was applied to a set of ground deformation data related to two consecutive surveys performed in July 2001 and September 2001. The

Table 9 The traditional Mogi and Okada models for generating synthetic SAR data

Type of the model	The model parameters and the figure	The related components u_x, u_y and u_z of the displacement
Mogi		$u_x = \frac{C}{2\mu}\left\{\frac{x}{R_1^3} - \frac{\lambda+3\mu}{\lambda+\mu}\frac{x}{R_2^3} - \frac{6xz(z+f)}{R_2^5}\right\}$ $u_y = \frac{C}{2\mu}\left\{\frac{y}{R_1^3} - \frac{\lambda+3\mu}{\lambda+\mu}\frac{y}{R_2^3} - \frac{6yz(z+f)}{R_2^5}\right\}$ $u_z = \frac{C}{2\mu}\left\{\frac{z-f}{R_1^3} - \frac{(\lambda-\mu)z-(\lambda+3\mu)f}{(\lambda+\mu)R_1^3} - \frac{6z(z+f)^2}{R_2^5}\right\}$ Where: $R_1 = \sqrt{x^2+y^2+(z-f)^2}\, R_2 = \sqrt{x^2+y^2+(z+f)^2}$ λ and μ are Lam`e's constant, c is the nucleus strain. Assuming that the radium is small compared with the depth f, the nucleus strain can be computed as: $C = -\frac{1}{2}a^3\Delta P$
Okada		$\begin{cases} u_x = \frac{U_3}{2\pi}\left[\frac{q^2}{R(R+\eta)} - I_3\sin^2\delta\right] \\ u_y = \frac{U_3}{2\pi}\left[\frac{-\bar{d}q}{R(R+\xi)} - \sin\delta\left\{\frac{\xi q}{R(R+\eta)} - \tan^{-1}\frac{\xi\eta}{qR}\right\} - I_1\sin^2\delta\right] \\ u_z = \frac{U_3}{2\pi}\left[\frac{\bar{y}q}{R(R+\xi)} + \cos\delta\left\{\frac{\xi q}{R(R+\eta)} - \tan^{-1}\frac{\xi\eta}{qR}\right\} - I_5\sin^2\delta\right] \end{cases}$ $\begin{cases} p = y\cos\delta + d\sin\delta \\ q = y\sin\delta - d\cos\delta \\ \tilde{y} = \eta\cos\delta + q\sin\delta \\ \tilde{d} = \eta\sin\delta - q\cos\delta \\ R^2 = \xi^2+\eta^2+q^2 = \xi^2+\tilde{y}^2+\tilde{d}^2 \\ X^2 = \xi^2+q \end{cases}$

(continued)

Table 9 (continued)

Type of the model	The model parameters and the figure	The related components u_x, u_y and u_z of the displacement
	Note that (1) both types of sources are supposed to work in a homogeneous and isotropic elastic half space. (2) Figures are redrawn after Nunnari et al. 2005	If cos(δ)#0: $\begin{cases} I_1 = \frac{\mu}{\lambda+\mu}\left[\frac{-1}{\cos\delta}\frac{\xi}{R+\widetilde{d}}\right] - \frac{\sin\delta}{\cos\delta}I_5 \\ I_2 = \frac{\mu}{\lambda+\mu}[-\ln(R+\eta)] - I_3 \\ I_3 = \frac{\mu}{\lambda+\mu}\left[\frac{1}{\cos\delta}\frac{\widetilde{y}}{R+\widetilde{d}} - \ln(R+\eta) + \frac{\sin\delta}{\cos\delta}I_4\right] \\ I_4 = \frac{\mu}{\lambda+\mu}\frac{1}{\cos\delta}\left[\ln\left(R+\widetilde{d}\right) - \ln(R+\eta)\sin\delta\right] \\ I_5 = \frac{\mu}{\lambda+\mu}\frac{2}{\cos\delta}\tan^{-1}\frac{\eta(X+q\cos\delta)+X(R+X)\sin\delta}{\xi(R+X)\cos\delta} \end{cases}$ If cos(δ) = 0: $\begin{cases} I_1 = -\frac{\mu}{2(\lambda+\mu)}\frac{\xi q}{\left(R+\widetilde{d}\right)^2} \\ I_3 = \frac{\mu}{2(\lambda+\mu)}\left[\frac{\eta}{R+\widetilde{d}} + \frac{\widetilde{y}q}{\left(R+\widetilde{d}\right)^2} - \ln(R+\eta)\right] \\ I_4 = \frac{\mu}{\lambda+\mu}\frac{q}{R+\widetilde{d}} \\ I_5 = \frac{\mu}{\lambda+\mu}\frac{\xi\sin\delta}{R+\widetilde{d}} \end{cases}$

Table 10 Two types of fitness function used in GA to invert SAR data

Type of fitness function	Related expression and		Concept
Least square error	$J = \sum_{i=1}^{N} \sqrt{(O_i - P_i)^2}$	Better results when N is relatively small (e.g. N = 30)	$RMSE = \frac{J}{\sqrt{N}}$
Index of disagreement	$J = \frac{\sum_{i=1}^{N} (P_i - O_i)^2}{\sum_{i=1}^{N} \left(\left(\lvert P_i - \overline{O} \rvert + \lvert O_i - \overline{O} \rvert \right)^2 \right)}$	More appropriate for N larger than 30	d = index of agreement = 1−J
Note that N represents the number of measuring points while O_i and P_i indicate the expected (observed) and simulated values, respectively			

Table 11 Performance indices computed to evaluate the error related to the GAs inverse modelling approach, note that bias is $\frac{1}{N}\sum (Oi - Pi)$, (Nunnari et al. 2005)

Parameter	Bias	RMSE	NMAE%	d
Dip angle (°)	−0.2533	0.3136	0.2815	0.9999
Strike angle (°)	−0.2435	0.3215	0.0676	0.9997
Length (m)	0.0177	0.0247	0.2952	0.9965
Width (m)	0.0027	0.0155	0.3291	0.9978
Strike Slip (m)	−0.0642	0.1561	3.4742	0.9990
Dip Slip (m)	0.1998	0.2117	4.9946	0.9904
Opening (m)	−0.0048	0.0210	0.5161	0.9972
Depth (m)	0.1206	0.1330	3.0139	0.9956
X_S (m)	−0.1455	0.1822	0.7275	0.9877
Y_S (m)	0.0178	0.2831	1.3329	0.9994

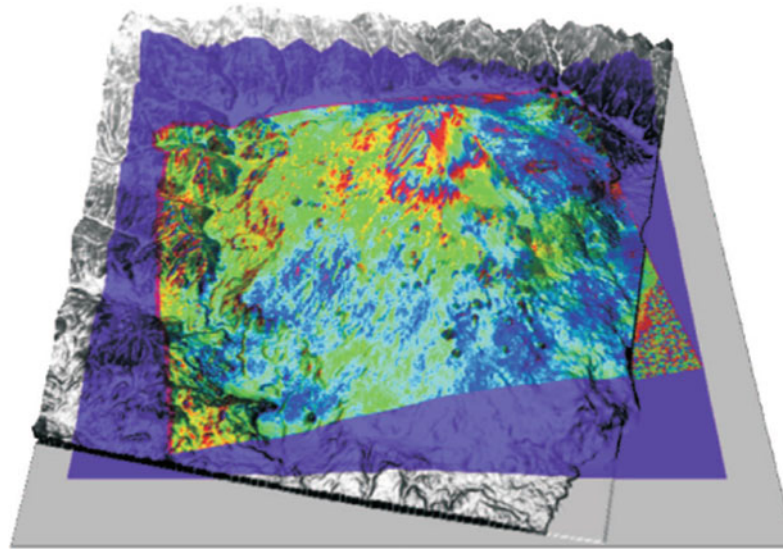


Fig. 7 Differential interferogram in phase (−Π, Π) relevant to Mt. Etna area referring to the ascending pair 22 July–26 August 1998 (Nunnari et al. 2005)

results showed an acceptable agreement among the Latitude, Longitude, Azimuth, Depth, Dip and Opening parameters, however there are notable differences in length and width values for GPS and SAR data inversion. A probable reason that Nunnari et al. (2005) mentioned to interpret this difference was that the SAR data set implies the presence of a deep volume in the basement,

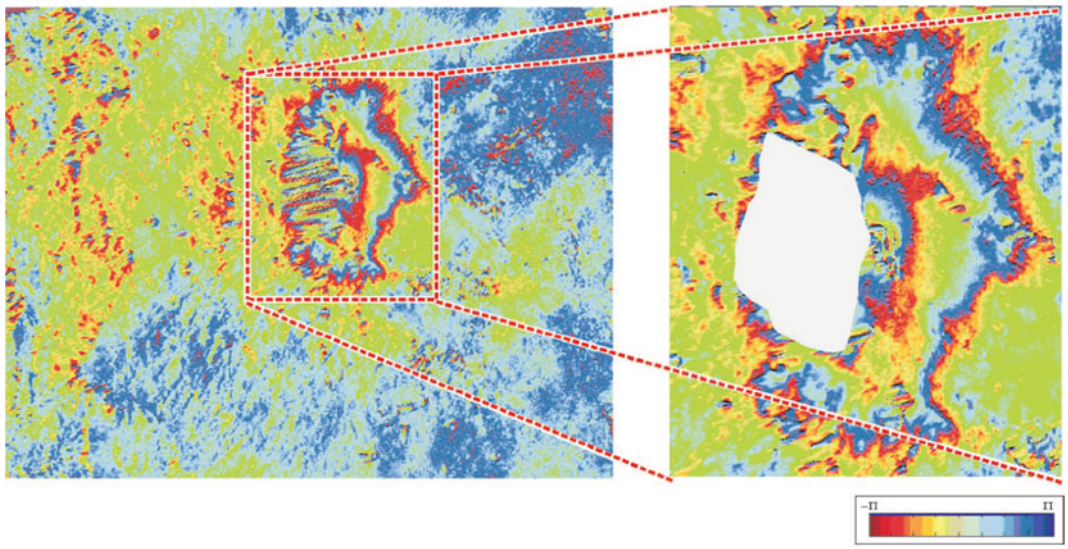


Fig. 8 (Left) Geocoded ascending interferogram refers to image pair 22 July–26 August 1998 and (right) considered detail (phase). The blank area shows the effect of a filter applied to mask the geometric errors that typically affect SAR interferograms (Nunnari et al. 2005)

Table 12 The results of GA for inversion of ground deformation through InSar data recorded at Etna between 22 July and 26 August 1998 using Mogi model (Nunnari et al. 2005)

Searched parameter	Returned value	Range search	
		Minimum	Maximum
−2C (GPa * m^3)	−47,600,000	−80,000,000	−10,000,000
f-Depth to reference level (m)	3151	10	3200
X_S—East coordinate (m)	500,316	494,000	504,000
Y_S—North coordinate (m)	4,178,936	4,172,500	4,184,500

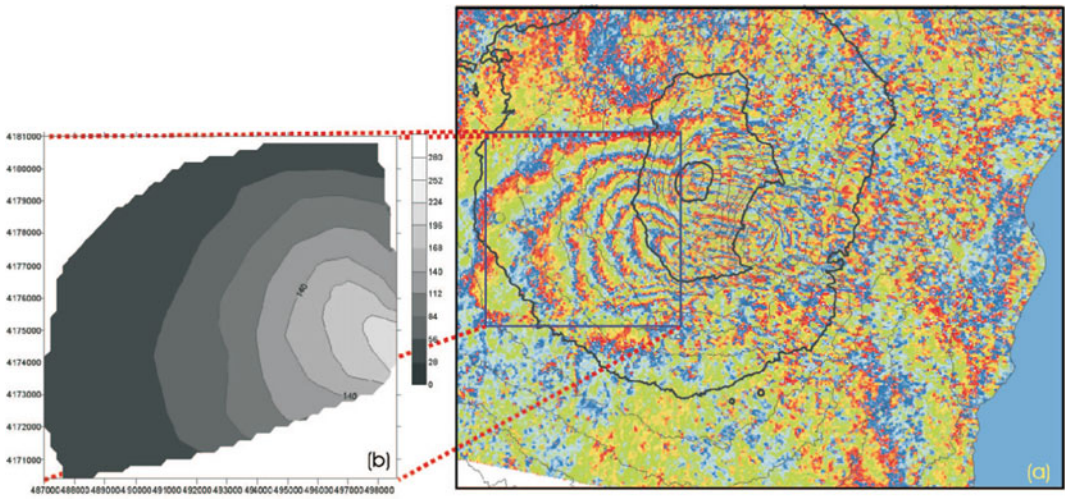


Fig. 9 **a** Geocoded descending interferogram referring to the image pair 15 November 2000–31 October 2001 of the Mt. Etna area, **b** geocoded LOS displacement map (expressed in mm) of resize area (Nunnari et al. 2005)

under the west flank of Etna which was pressurizing before the main eruption and depressurizing rapidly during the eruption.

4 Automatic Monitoring System of Infrasonic Events at Mt. Etna Using Genetic Approach

4.1 The Infrasonic Events Automatic Monitoring

Montalto et al. (2010) used genetic algorithm for estimation of source location and modelling of volcano-infrasonic events. They applied automatic procedures to the data recorded through a permanent network consists of five infrasound stations located at Mt. Etna volcano from 1.5 to 7 km distance to summit area center, running by INGV, Section of Catania (Fig. 10), in order to achieve a real-time implementation. The infrasound signals at Mt. Etna consist in amplitude transients, called infrasound events.

The comprehensive system of infrasonic events automatic monitoring consists of the following parts (Montalto et al. 2010):

i. data acquisition,
ii. event detection,
iii. event characterization,
iv. source location and finally,
v. modelling.

These parts are depicted in Fig. 11 with more details, as it is shown this automated monitoring system has two different part: online parts (i–iv) which is for real time application and off-line (v) that is designed for near-real time analysis. The specifications of the infrasonic sensors and the details of data acquisition is shown in Table 14.

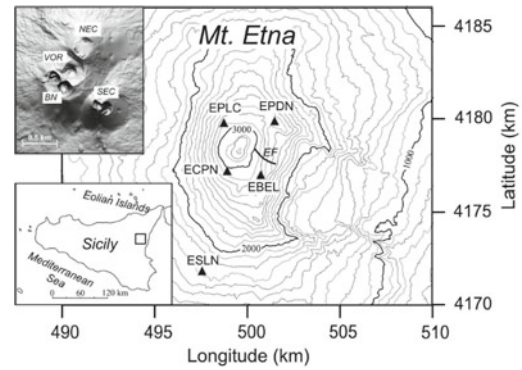


Fig. 10 Map of the summit area of Mt. Etna with the location of the five infrasonic sensors (triangles), composing the permanent infrasound network, and the eruptive fissure opened on May 13, 2008 (black line "EF"). The digital elevation model in the upper left corner shows the distribution of the four summit craters (VOR = Voragine, BN = Bocca Nuova, SEC = South-East Crater, NEC = North-East Crater) (Montalto et al. 2010)

4.2 Genetic Algorithm Method for Infrasonic Source Parameters Estimation

Following the schematic shown in Fig. 11, when the waveforms of infrasonic events is characterized, and the source location is determined, the mechanism of the can be investigated. Among the available infrasound source models, three of

Table 13 GA results for inverting independent SAR and GPS data (Nunnari et al. 2005)

Searched parameter	Inversion with SAR data	Inversion using GPS data
Lat. UTM (km)	500.77	500.65
Long UTM (km)	4176.275	4175.5
Azimuth (°)	1.30	3.40
Depth (km)	1.5 a.sl	1.6 a.sl
Length (km)	7.40	2.34
Width (m)	1.50	3.55
Dip (°)	89.90	89
Opening (m)	3.00	2.51

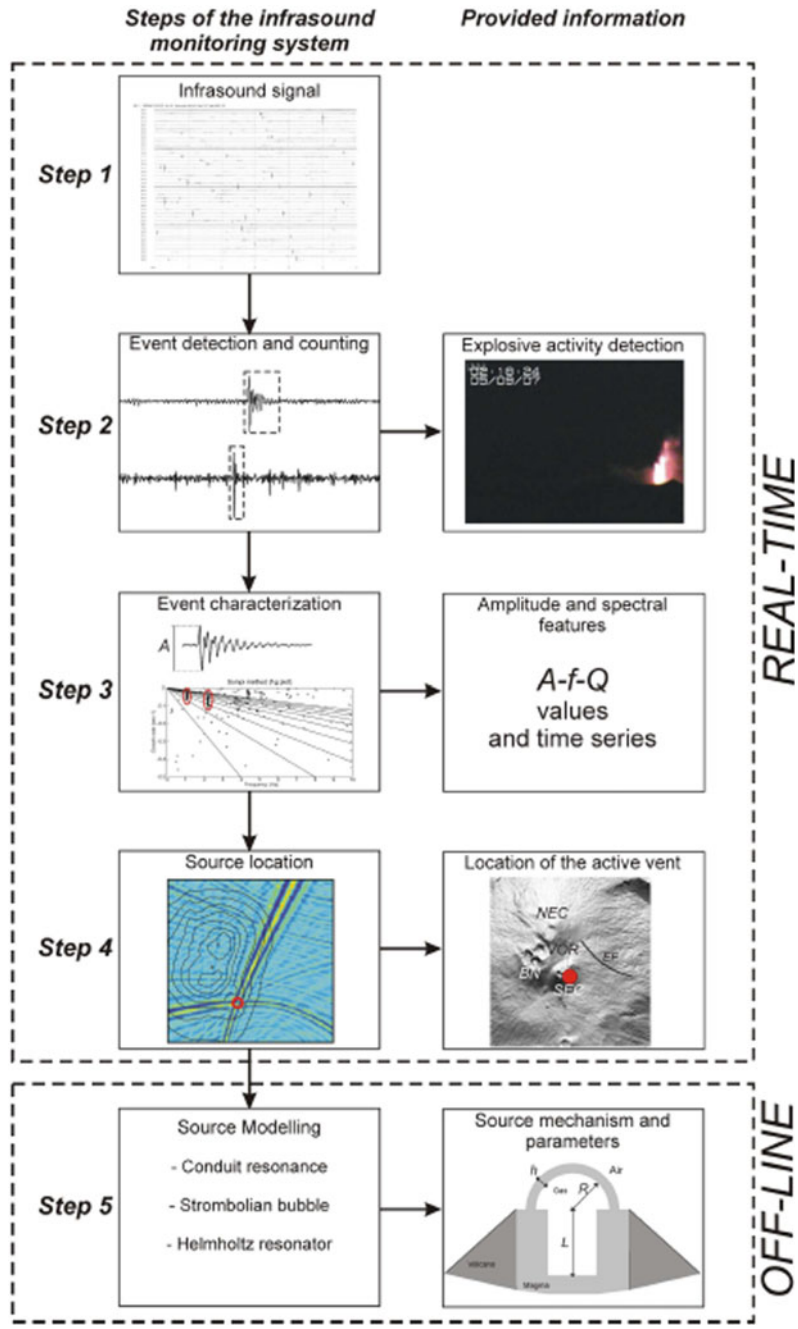


Fig. 11 Schematic diagram of the automatic infrasound monitoring system (Montalto et al. 2010)

them are mostly common to apply on observed data, listed in Table 15 The related forward modeling equations and formulation for infrasonic source models is a long story explained with great details in Montallo et al. (2010) which is beyond the scope of this book and the interested readers are referred to the mentioned article for more details.

3. These models were used to generate synthetic data. The sketch of Strombolian bubble vibration and Helmholtz resonator model are depicted in Fig. 12.

Table 14 Infrasonic sensors specification

Type of infrasonic sensor	Monacor condenser microphone MC-2005
Sensitivity of the sensor	80 mV/Pa in the 1–20 Hz infrasonic band
Infrasonic signals transmission mode	In real-time by radio link
Data acquisition receiver location	Catania center
Sampling rate of gathering data in acquisition center	100 Hz

The problem of identification is formulated as a type of optimization task to find the parameters that minimize the estimation error between real data and that of GA output (Fig. 13). The GA inversion was considered to be completed when the prediction error was less than a threshold or if a time-out condition happened. In the former condition the source parameters were saved in a suitable database but in the latter the related event was discarded (Fig. 13).

4.3 Evaluation of the Proposed Method

To evaluate the GA inversion efficiency, a record sample of EBEL station was selected. This station was also considered as reference station at Etna because it had the best signal to noise ratio among all infrasonic stations. GA results were investigated for both Strombolian bubble vibration and Helmholtz resonator models. After the model parameters were optimized through GA method the related synthetic infrasonic data was calculated and compared with observed infrasonic waveforms (Fig. 14). As it can be seen the infrasonic waveforms calculated via GA outputs are in good agreement with the real data.

5 Rapid Estimation of Earthquake Magnitude and Source Parameters Using Genetic Algorithms

Novianty et al. (2021) presented a rapid estimation method to estimate the magnitude of earthquakes and the source parameters through GA, the method was tested on the 2011 Tohoku Oki earthquake. Through their method magnitudes and geometrical parameters of the earthquake source are estimated using GPS time series which are recorded during an earthquake event. Here, GA is used to estimate the parameters to invert the elastic dislocation equations based on Okada's model, as this model was presented in detail in Sect. 2.2 earlier, we have avoided duplicating it.

The GA hyperparameters have been tuned to get the best configuration settings that provide the best parameter estimates. The selected configuration was then tested using actual displacement time-series data from GPS stations affected by 2011 off the Pacific coast of the Tohoku Earthquake. Novianty et al. (2021) processed GPS time series of all stations to estimate permanent shift values for both horizontal and vertical components. The estimated permanent displacement was then inverted through GA to find the optimized source parameters in the large space of Okada's model parameters.

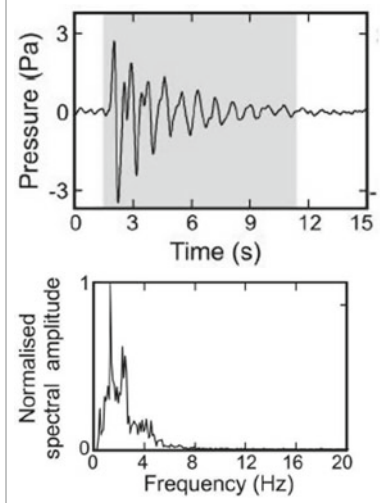
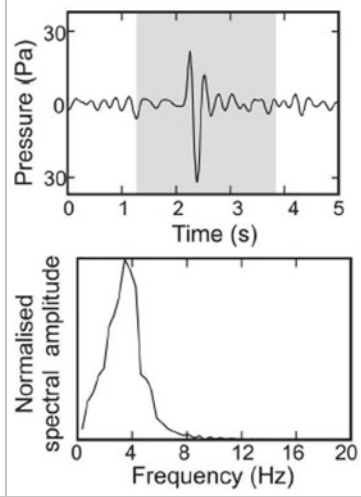
5.1 Displacement Detection and Estimation

To estimate the permanent displacement that occurs at certain observation points a characteristic function (D) was used which is a function of parameters of Short-Term and Long-Term Average, namely STA(t) and LTA(t) respectively, defined as follows:

$$D = |STA(t) - LTA(t)| - SD(LTA(t)) \quad (7)$$

where:

Table 15 The most common infrasound source models used to generate synthetic data

Source model	Reference(s)	Description	Example of candidate infrasonic signal (after Montalto et al. 2010)
Resonating conduit model	Buckingham and Garces (1996), Garces and McNutt (1997), Hagerty et al. (2000)	Based on pipe-like conduit, The acoustic signal, thus generated, consists in gradually decaying sinusoids with a fundamental mode and harmonics	
Strombolian bubble vibration	Vergniolle and Brandeis (1994, 1996), Vergniolle et al. (1996, 2004), Vergniolle and Ripepe (2008)	Produced by the vibration of a thin layer of magma, pushed by a variation of pressure inside a shallow metric bubble prior to bursting	
Helmholtz resonator model	Vergniolle and Caplan-Auerbach (2004)	Local coalescence within a magma foam	Similar to the signal types shown in reasoning conduit row of this table

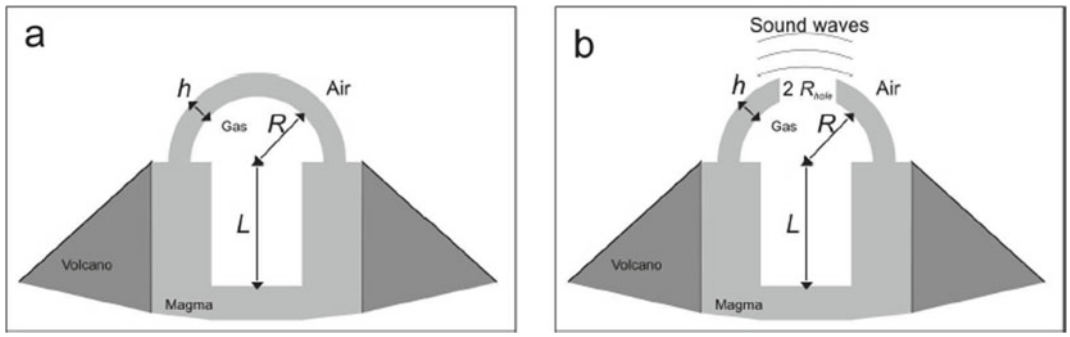


Fig. 12 **a** Sketch of a vibrating bubble and **b** The Helmholtz resonator; at the top of a magma column (in Montalto et al. 2010, redrawn after Vergniolle et al. 2004; Vergniolle and Caplan-Auerbach 2004)

$$\mathrm{STA(t)} = \frac{\sum_{i=t-\propto+1}^{t} (p_i.x_i)}{\sum_{i=t-\propto+1}^{t} p_i}, LTA(t) = \frac{\sum^{t} \sum_{i=t-\beta+1}^{t} (p_i.x_i)}{\sum_{i=t-\beta+1}^{t} p_i} \quad (8)$$

where x_i is the norm of the horizontal components at t = i, α and β are the suitable time window lengths based on the noise properties of the real-time kinematic GPS time series (this study assumes α = 60 s and β = 600 s), and pi is the weighting parameter (Novianty et al. 2021).

D is a key value for detecting permanent displacement. Permanent displacement is said to occur and be detected when the value of D passes a certain threshold value. The threshold used is expressed in K, which is defined as follows:

$$\mathrm{K} = 4.\mathrm{SD(D)} \quad (9)$$

where SD(D) is the standard deviation of D.

Permanent displacement detecting and estimating flowcharts are shown in Fig. 15.

5.2 Moment Magnitude (Mw) Estimation

Moment is depending directly on the total of the energy which is released during an earthquake and formulated as:

$$\mathrm{Mo} = \mu.\,\mathrm{Area.\,slip} = \mu.(\mathrm{L.\,W}).\,\mathrm{Ls} \quad (10)$$

where Mo is moment seismicity, μ is the rigidity parameter or shear modulus of the crust (approximately 3–1010 N/m), L is the fault length (m), W is the fault width (m), and L_s is the slip length (m).

Using the Eq. (11) "Moment seismicity: Mo" can be converted into moment magnitude of the related earthquake:

$$Mw = \frac{2}{3}.\log(Mo) - 6.06 \quad (11)$$

5.2.1 GA-Okada Procedure

Reference to Okada model:

Okada(Latitude, Longitude, depth, strike, dip, length, width, rake, slip) = [uE, uN, uZ]

Where [uN,uE,uZ] are displacement values at the observation points in the directions of north (N), east (E), and vertical (Z), respectively.

So displacement estimation (distest) can be calculated using the forward Okada's model as follows:

distest = Okada(Latitude, Longitude, depth, strike, dip, v_1, v_2, v_3, v_4)

where Latitude, Longitude, depth, strike, and dip are the predetermined parameters reference to the earthquake's fault model used in the calculation as described in Table 8, also v_1, v_2, v_3, v_4 are the estimated variables of the source parameter length, width, rake, and slip, respectively by GA. On the other hand, GA optimizes the four last mentioned parameters (Novianty et al. 2021).

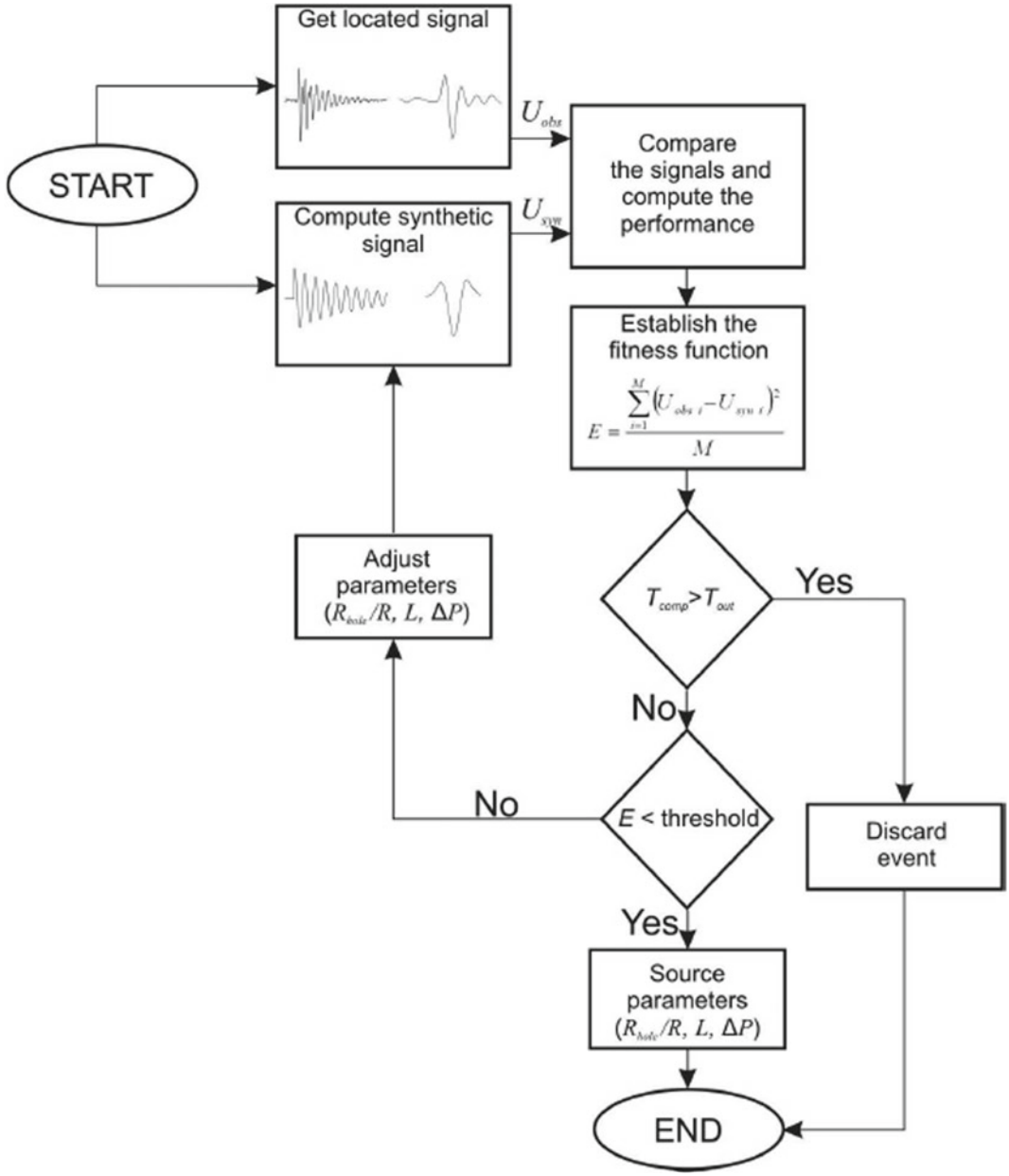


Fig. 13 The flowchart of parameter estimation by Genetic Algorithm (GA). Note that Tcomp and Tout indicate the computational time and the fixed time-out, respectively (Montalto et al. 2010)

5.2.2 Evaluation for Real Data

Novianty et al. (2021) evaluated the GA-Okada model accuracy and efficiency using two cases in order to invert the earthquake displacement into the length, width, rake, and slip of the related fault as the source of the quake. In the first case 737 GPS stations, related to the 2011 Tohoku Oki earthquake, were as the observation points. Multiple fault models were used in the experiments to obtain valid results for GA performance. The four predetermined seismic fault models used at this stage of the experiment are

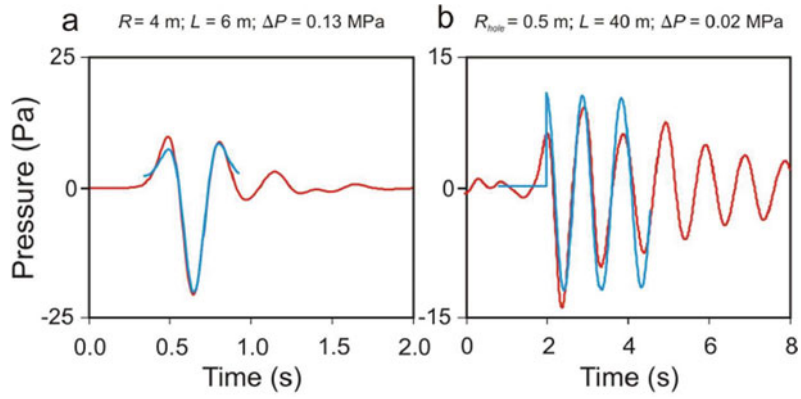


Fig. 14 Comparison between the observed waveforms of infrasonic events recorded by EBEL station (red) and the synthetic ones (blue) calculated via GA model parameters estimation considering **a** Strombolian bubble vibration and **b** Helmholtz resonator models (Montalto et al. 2010)

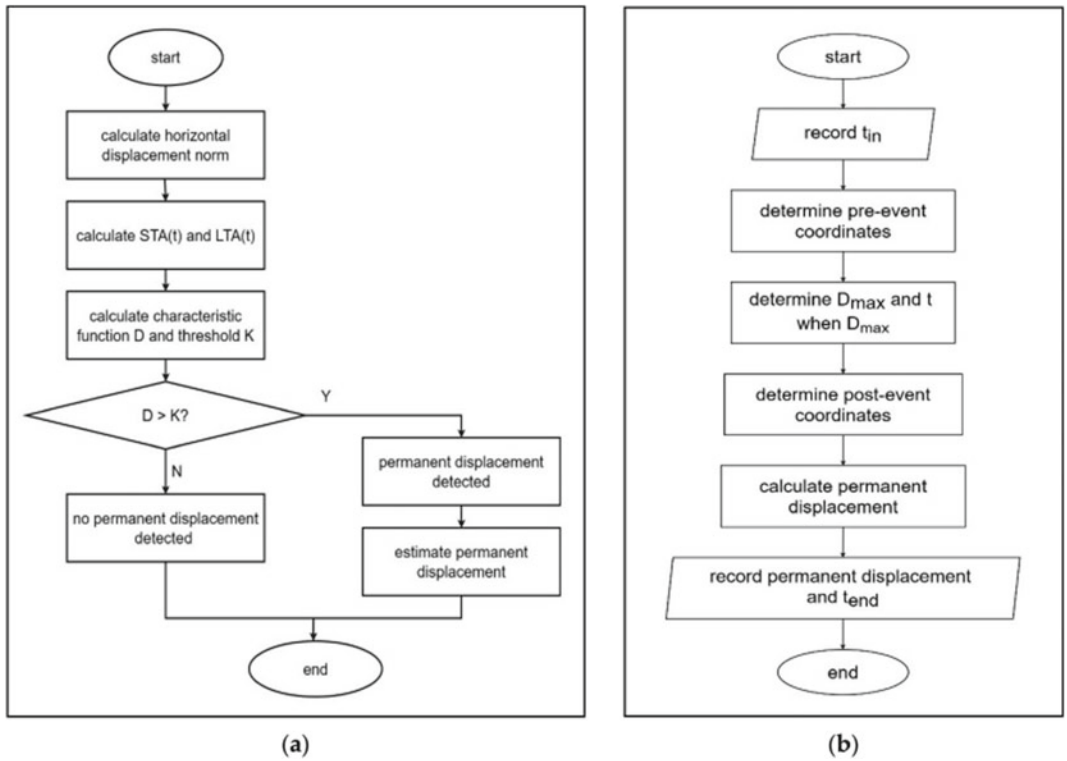


Fig. 15 Flowchart of the algorithm: **a** permanent displacement detection; **b** permanent displacement estimation

shown in Table 16 and the observation points, focal mechanism of the 2011 earthquake, and aftershocks is depicted in Fig. 16. Also, the hyperparameter values in the tuning of GA is listed in Table 17.

In Fig. 17 the comparison of GA displacement with Okada's displacement test, for all four fault models characterized in Table 15, is shown. Blue vectors represent GA displacement while black vectors represent Okada's displacement.

Table 16 Earthquake fault models used for 2011 Tohoku Oki earthquake

Parameters	Model 1	Model 2	Model 3	Model 4
Lat. (°)	142.834	142.834	144.00	142.80
Long. (°)	38.17	38.17	38.80	37.33
Depth (km)	20	21	5.1	17
Strike (°)	210	201	203	203
Dip (°)	9	9	16	15
Length (km)	250	625	186	194
Width (km)	50	280	129	88
Rake (°)	90	104	101	83
Slip (m)	2	6	24.70	6.1

Coinciding vectors implies that the estimation results are very closer to the reference value. The centroid of each fault model is indicated by a red pentagram.

The second case used for testing of Okada-GA was 2018 Lombok Earthquake Event. The time series data used for the test was real-time 1 Hz kinematic GPS data recorded at the stations

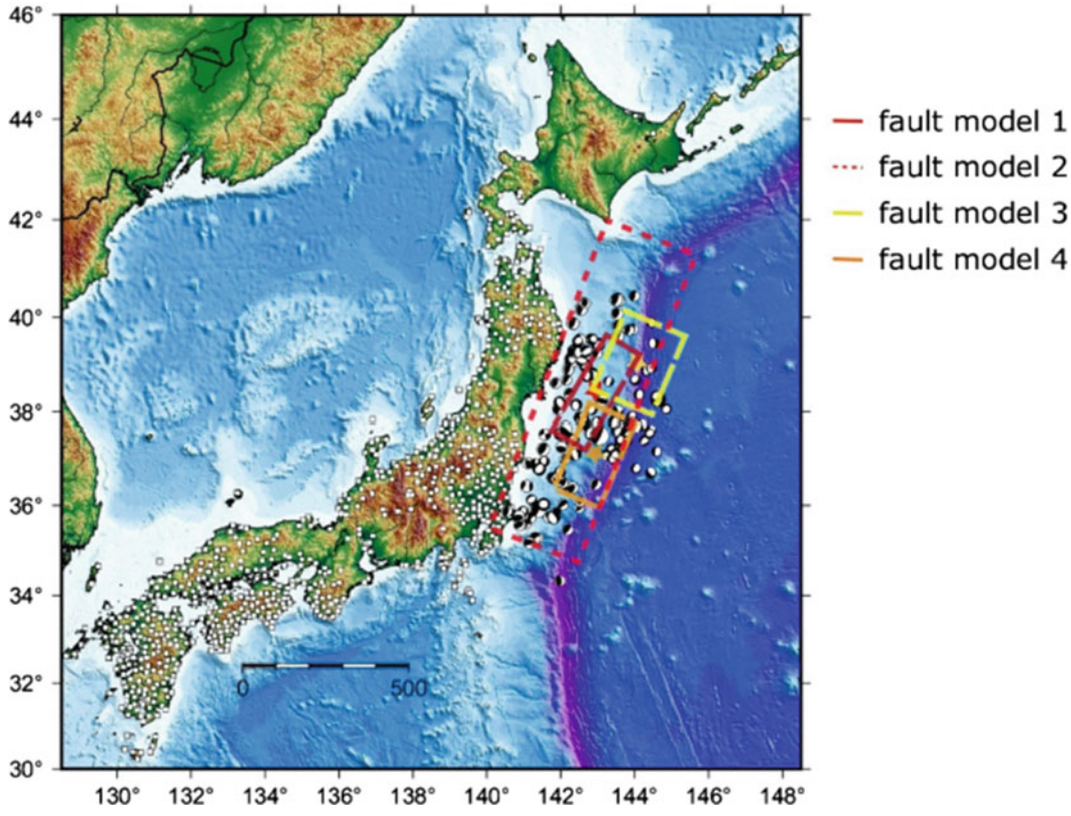


Fig. 16 Distribution map of observation points with focal mechanism of the 2011 Tohoku Oki earthquake, and related aftershocks (Novianty et al. 2021)

Table 17 Hyperparameter values in the tuning of Okada-GA

List of values	Description	Hyperparameters
[100, 200, 500]	Number of generations	*n_gen*
[20, 40]	Size of the population	*n_pop*
[16, 20, 24]	Number of chromosome bits	*n_bits*
[0.8, 0.9]	Crossover rate	*r_cross*
[1.563, 1.250, 1.042]	Mutation rate ($\times 10^{-2}$)	*r_mut*

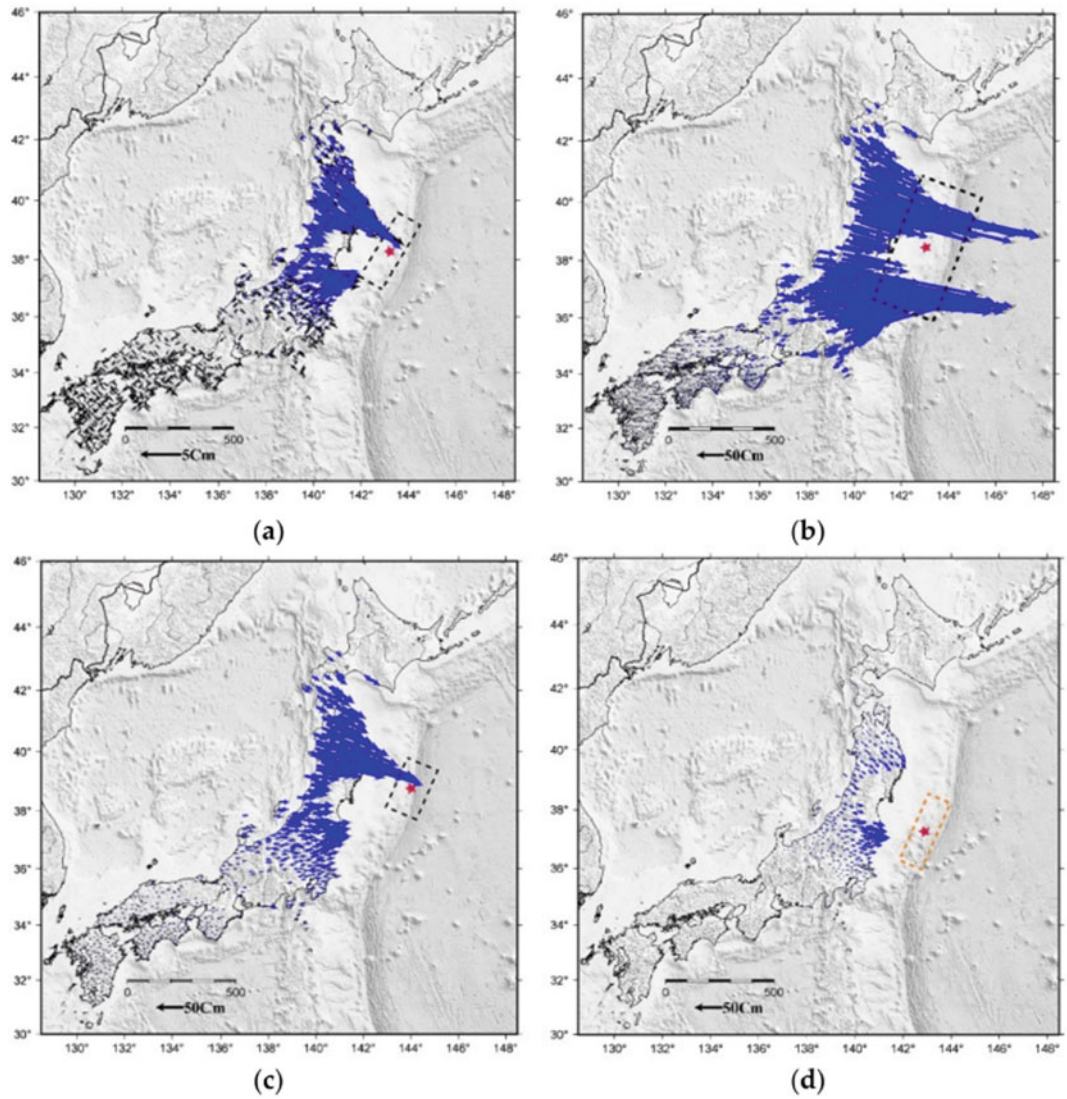


Fig. 17 Comparison of the displacements between the estimation (GA displacement) and the reference displacement (Okada's displacement): **a** fault model 1; **b** fault model 2; **c** fault model 3; **d** fault model 4 (Novianty et al. 2021)

during this major earthquake. A comparison of the GA displacement and the reference permanent displacement of the 2018 Lombok seismic event is depicted in Fig. 18. GA displacement and reference displacement are shown by a blue vector and a black vector, respectively. The red

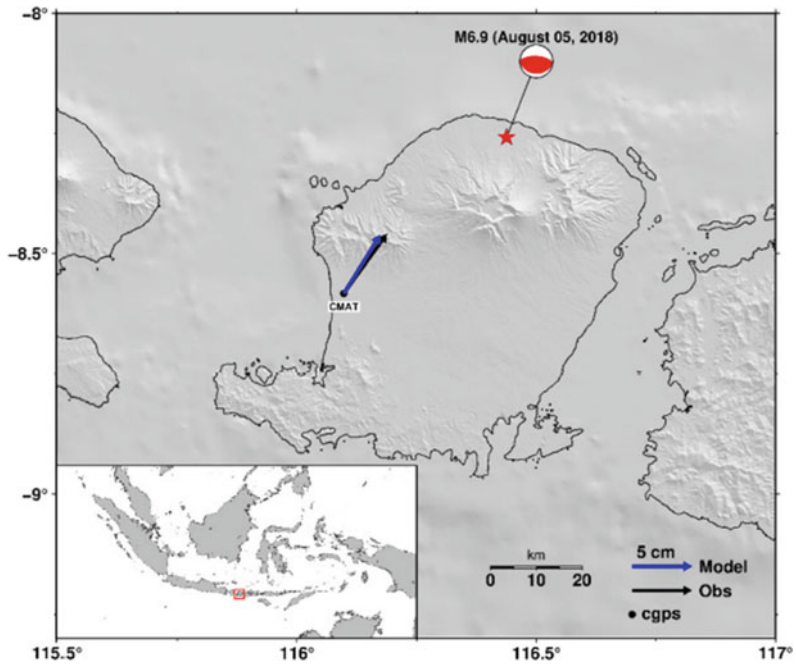


Fig. 18 Comparison of GA displacement and referenced permanent displacement of the 2018

pentagram is the indicator of the Lombok Earthquake 2018 epicenter. The estimated Mw result was 6.898 which is very close to classical methods calculations.

Lombok earthquake (Novianty et al. 2021).

6 Generator of Genetic Seismic Signals

6.1 Introduction

García and Alcántara (2019) used GA to assemble time series of accelerations likeminded with a prescribed goal spectrum. They presented a generator of seismic indicators which takes into consideration both the earthquake and the soil characteristics for a realistic depiction of ground motions. The proposed method uses GA to convert the time series iteratively. Mimicking mating, natural choice, and mutation, the generator of genetic seismic indicators, modifies the accelerations additives of information considered appropriate parents in order to produce a success individuals or new most useful signals that excellent suit the goal situations. The procedure is immediately and consistent with excellent outcomes in facts that healthy a huge form of target spectra with minimum deviation even as keeping the seismological and geotechnical inherent traits of parents.

6.2 The Underlying Idea

There are two main techniques for scaling time histories shown in Fig. 19. In any case of the strategy domain, the forms of selecting the "initial" ground motions and their scaling to coordinate the design spectrum are partitioned and particular.

Utilizing GA to scale earthquake ground movements for plan is a continuation of such applications and parallels the appealing utilize of neural networks to accomplish the same errands (Ghaboussi and Lin 1998; Kim and Ghaboussi 1999). In order to use GA as a tool of transforming earthquake ground motions to match a

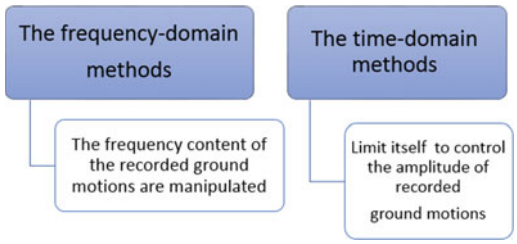


Fig. 19 Classification of techniques for scaling time histories

Table 18 Definition of the parameters of the fitness function in equation

T	The vibration period
$SA_{gi}(T)$	Spectral acceleration of genetic number at period
$SA_i(T)$	Spectral acceleration of target record number at period
T_i	Initial period to consider
T_f	Final period to consider

specific response spectrum, a number of accelerograms recorded at seismological stations, $[x_1, x_2,\ldots,x_n]$, is selected belonging to the input space *X*. In fact, each accelerogram is *a chromosome* and the set of *chromosomes* is designated as a *colony* or *population*. The fitness function, which is utilized to discover the leading combination of time arrangement that minimizes the distinction between a given target range and the one gotten from the genetically generated accelerograms, is defined as follow:

$$Z = \min\left\{\frac{1}{T_f - T_i}\sum_{i=1}^{f}\left(\left[SA_{gi}(T)\right] - \left[SA_i(T)\right]\right)^2\right\} \tag{12}$$

where the definition of all parameters in Eq. (12) are listed in Table 18.

The initial chromosomes were chosen according to geotechnical and seismic characteristics. The complete population was contained within the Mexican strong motion database which had more than 13,000 ground movement records (SMIS, 2000). Clearly, this populace is suitable for considering seismogenesis for the case study of Mexican subduction zone (García and Alcántara 2019).

Genetic seismic generator initials first with a randomly selection of parents (SE, M, FD, ST). The client can characterize from the endless universe of parents, which are listed in Table 19 and their conditions are more relevant for a specific examination. Then the fitness is calculated (equation), after that GA operators acts to produce new children and again the generating population is applied until the stop criteria (adaption) is met. The flowchart of the work is shown in Fig. 20.

6.3 Evaluation of GA Seismic Generator

In order to evaluate the GA method two cases were investigated both constrained by prescribed

Table 19 Geotechnical and seismic characteristics used as population in GA

Population members	Description	
Seismic environment (SE)	One of the related dynamic maps of México (see Fig. 21)	
Magnitude (M)	Mw	
FD	Focal depth	
Soil type (ST) in the recording station	Type A	For soft materials with high plastic index, high compressibility, high water content, and low to very-low shear wave velocities
	Type B	Deposits made up of stiff materials with high strength, high to very-high shear wave velocities and low to very-low compressibility potential
Note that on the off chance that the client does not have a priori seismic or geotechnical inclinations, the genetic generator chooses the beginning populace by random		

Table 20 The values of Q factors in Eq. (12) used to calculate final parents' quality in GA

Q factor	Condition	Value
Qgap gap is the largest azimuthal separation in degrees between two adjacent first-motion polarities projected on the Lambert–Schmidt equal-area projection circle	gap $\leq$ 180	(180 − gap)/90
	gap > 180	0
Qfitness	fit $\geq$ 0.7	(fit − 0.7)/0.15
	fit < 0.7	0
Qreadings Nr is the total number of polarity readings, and Nup is the number of compressional polarity readings	10 $\leq$ Nr $\leq$ 50	(Nr − 10)/20
	Nr $\leq$ 10	0
	Nr $\geq$ 50	2
Qpolarity	–	0.5 − (abs[(Nup/Nr − 0.5]/0:25)

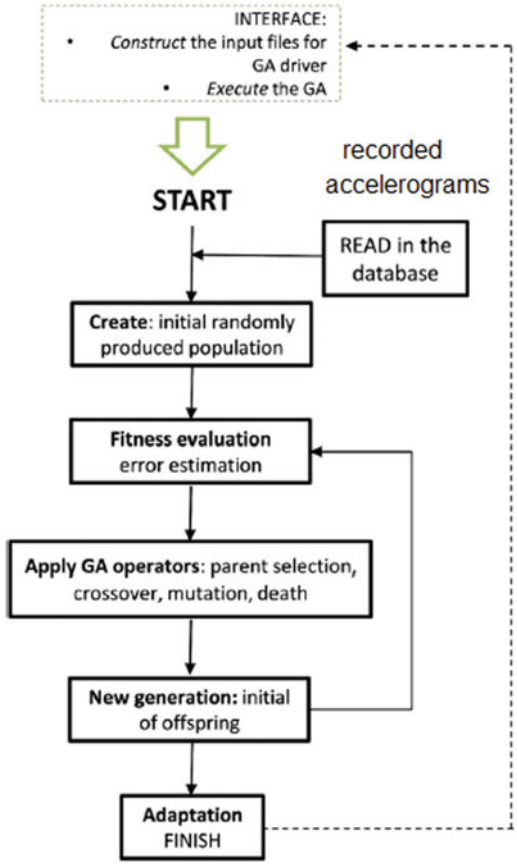


Fig. 20 The flowchart of GA seismic generator (redrawn after García and Alcántara 2019)

source and soil-type. In the first case, the type of soil was assumed A and the seismicity from zone 1, recorded magnitudes between 6 and 7 ($6 < M_w < 7$) with no initial estimation for focal depth (FD). The beginning populace was set at 100. The target Peak Ground Acceleration (PGA) was set break even with to 0.028 g. After 2780 generations, the stop criteria were accomplished and the spectra from the offspring can be assessed. The results are shown in Fig. 22. The great match shows the precision of the proposed procedure and the appropriateness of the created artificial accelerograms for exact plan purposes (García and Alcántara 2019).

In the second case the type of soil was B while seismicity from Zone 2 (as shown in Fig. 21), magnitude $5 < M < 6.5$, focal depth $10 < FD < 35$ km and initial population of 50 individuals. The created spectral accelerations through GA attain marginally higher values than the target spectrum at a few frequencies, however it lies exceptionally close the target around the natural frequency presented in Fig. 23 (García and Alcántara 2019).

7 Focal-Mechanism Determination in Taiwan by Genetic Algorithm

7.1 The Study Region

The focal mechanism of earthquakes, that their magnitude was greater than or equal by 4(ML 4) and happened in the region of Taiwan during 1991–2005, was determined through a GA

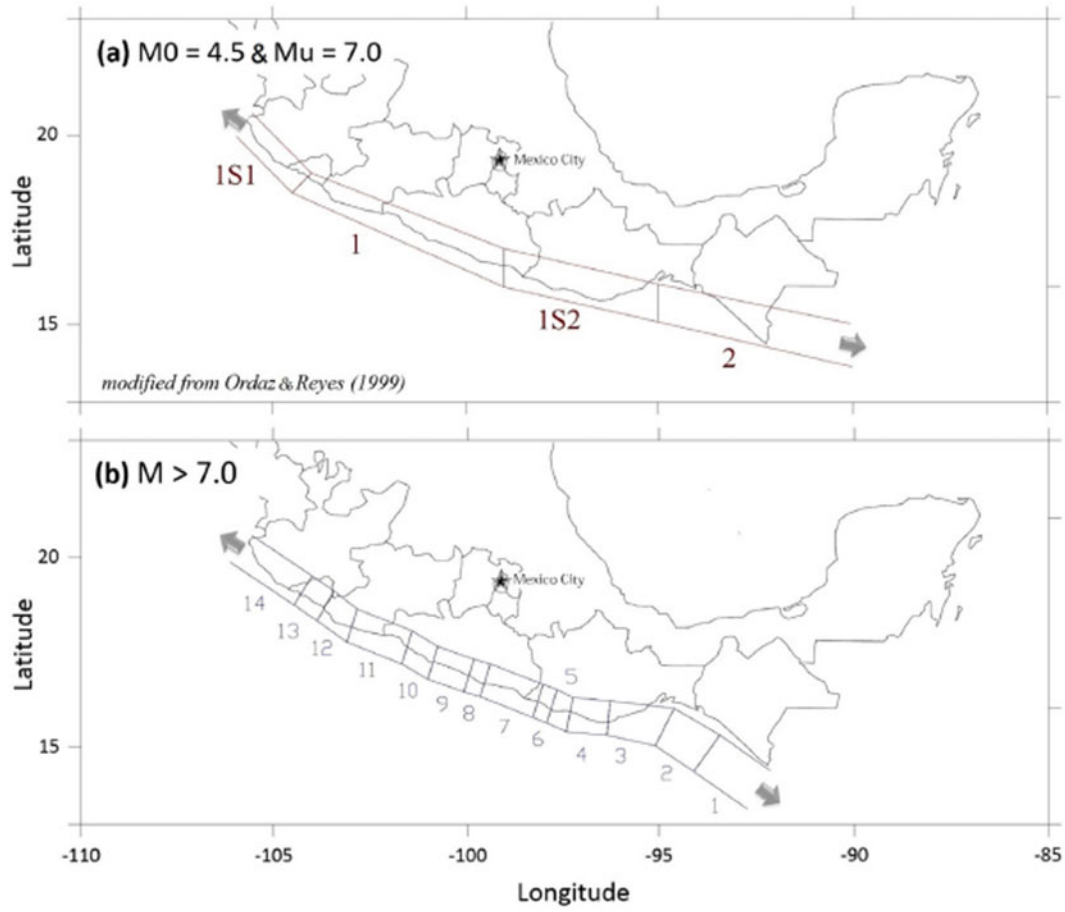


Fig. 21 Two seismic environments of Mexican subduction zone (García and Alcántara 2019)

approach (Wu et al. 2008). In this way, P waves first motion polarities that recorded at more than 700 seismic stations, located in Taiwan, were used (Fig. 24).

7.2 GA Procedure for Focal Mechanism Determination

Wu et al. (2008) used GA as an optimization tool to find the best focal mechanism based on the flowchart shown in Fig. 25. In their GA, a three-point cross over was used and the quality of the solutions were investigated through Eq. (12). They assigned Qfp = 0 for a no-constraint solution, and Qfp > 1 for a good solution. Some examples of focal mechanisms having different qualities are shown in Fig. 26.

$$\mathrm{Qfp} = \mathrm{Qgap} \cdot \mathrm{Qfitness} \cdot \mathrm{Qreadings} \cdot \mathrm{Qpolarity} \tag{12}$$

7.3 Test of GA for Synthetic Data

About 1000 synthetic events were used to find the optimal parameters of the proposed GA in order to provide a reliable focal mechanism solution. The parameters of GA that through this

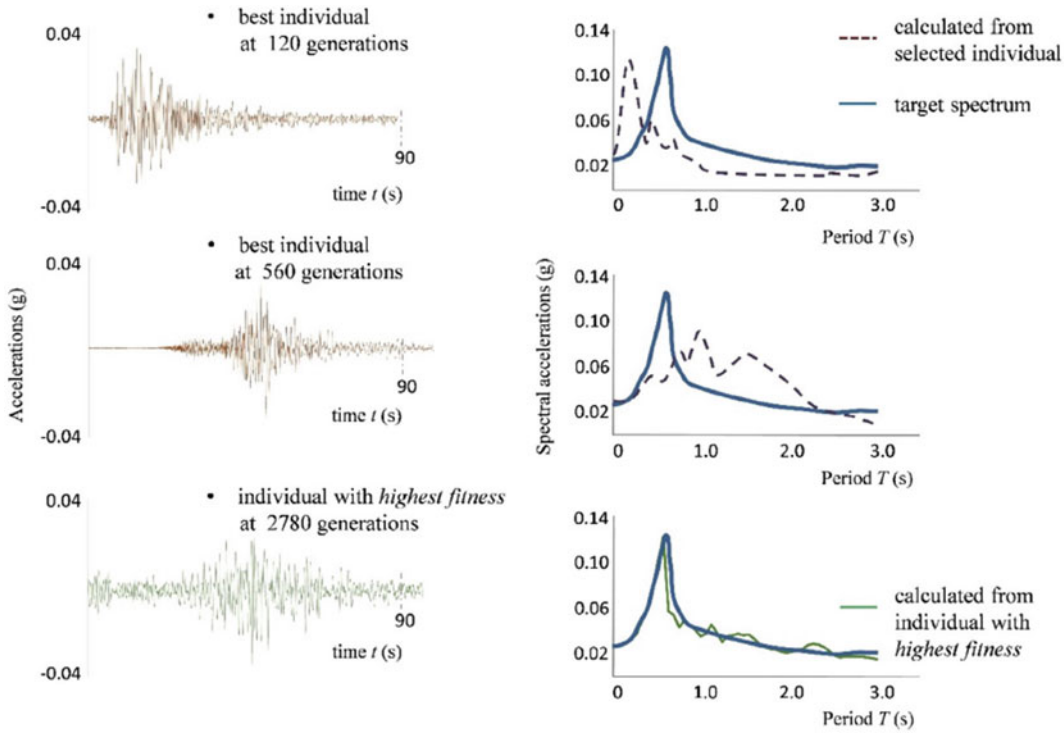


Fig. 22 The GA seismic generator results for Type B soil (García and Alcántara 2019)

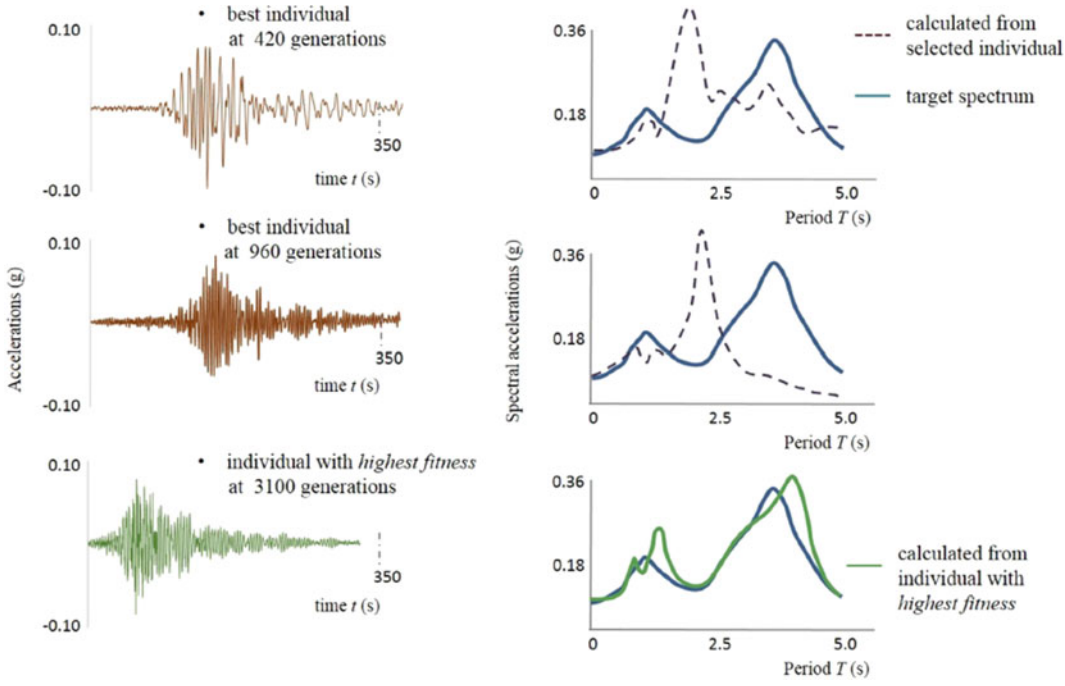


Fig. 23 The GA seismic generator results for Type A soil (García and Alcántara 2019)

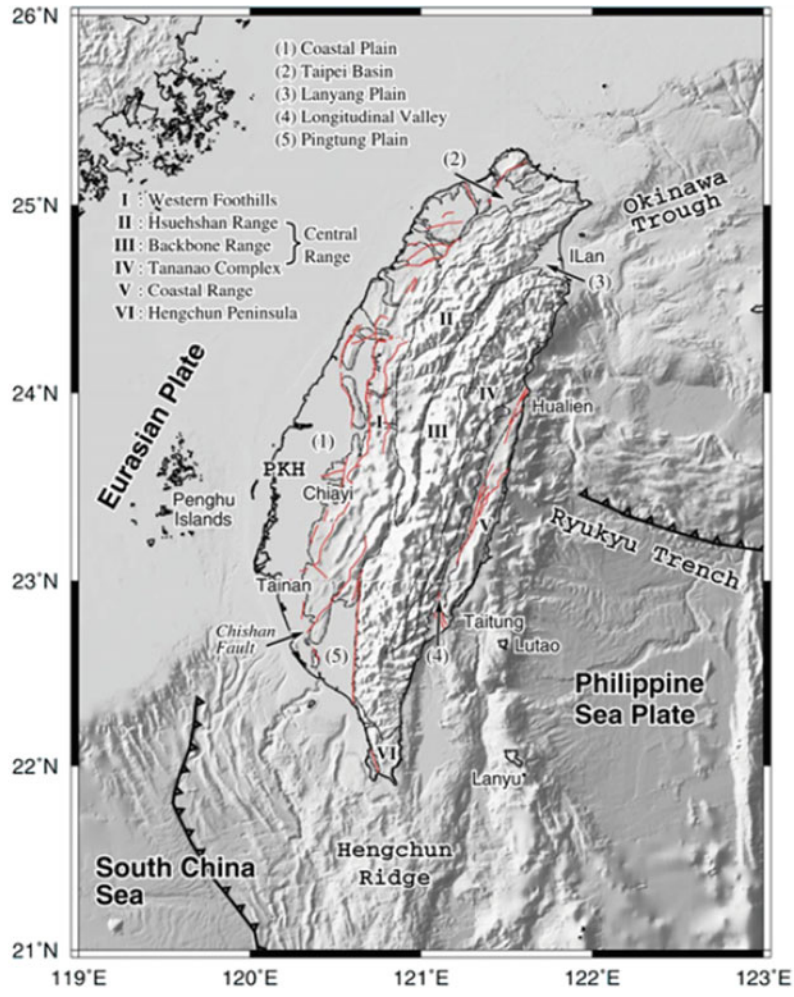


Fig. 24 Location of the active faults in Taiwan with its geological settings (Wu et al. 2008)

test were determided are population size, reproduction rate, number of bits for reversing in mutation process, mutation rate and pure cross over rate (Wu et al. 2008). The results for this test, search time for various values of the mentioned parameters, is shown in Fig. 27. Finally, the optimal parameters for GA were found which are listed in Table 21.

7.4 Test for Real Data

The proposed genetic algorithm was also evaluated by determination of first-motion focal mechanisms of earthquakes with magnitudes greater than or equal to four in the Taiwan region.

About 1635 earthquakes with their first-motion polarities were used to determine the related focal mechanisms. Totally the feature show that focal mechanism is the thrust-type which reflects the regime as a compressive stress type. Some of the comparisons of focal mechanisms determined through GA and the Harvard, USGS (Sipkin), and BATS solutions are shown in Fig. 28. Reference to the complex colliding plates in the region and its vicinity the results are in good agreement to other classical methods.

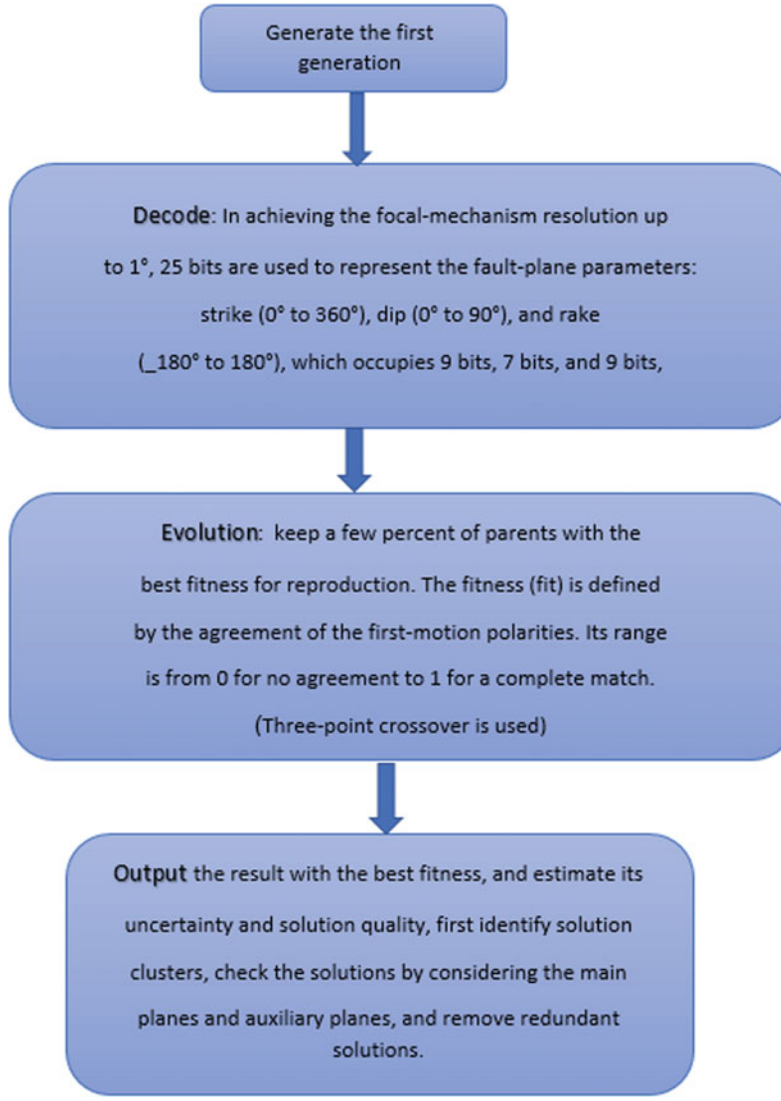


Fig. 25 Framework of GA solution for determining focal mechanisms (the solution quality is determined through Eq. (12), see Table 20) (extracted from Wu et al. 2008)

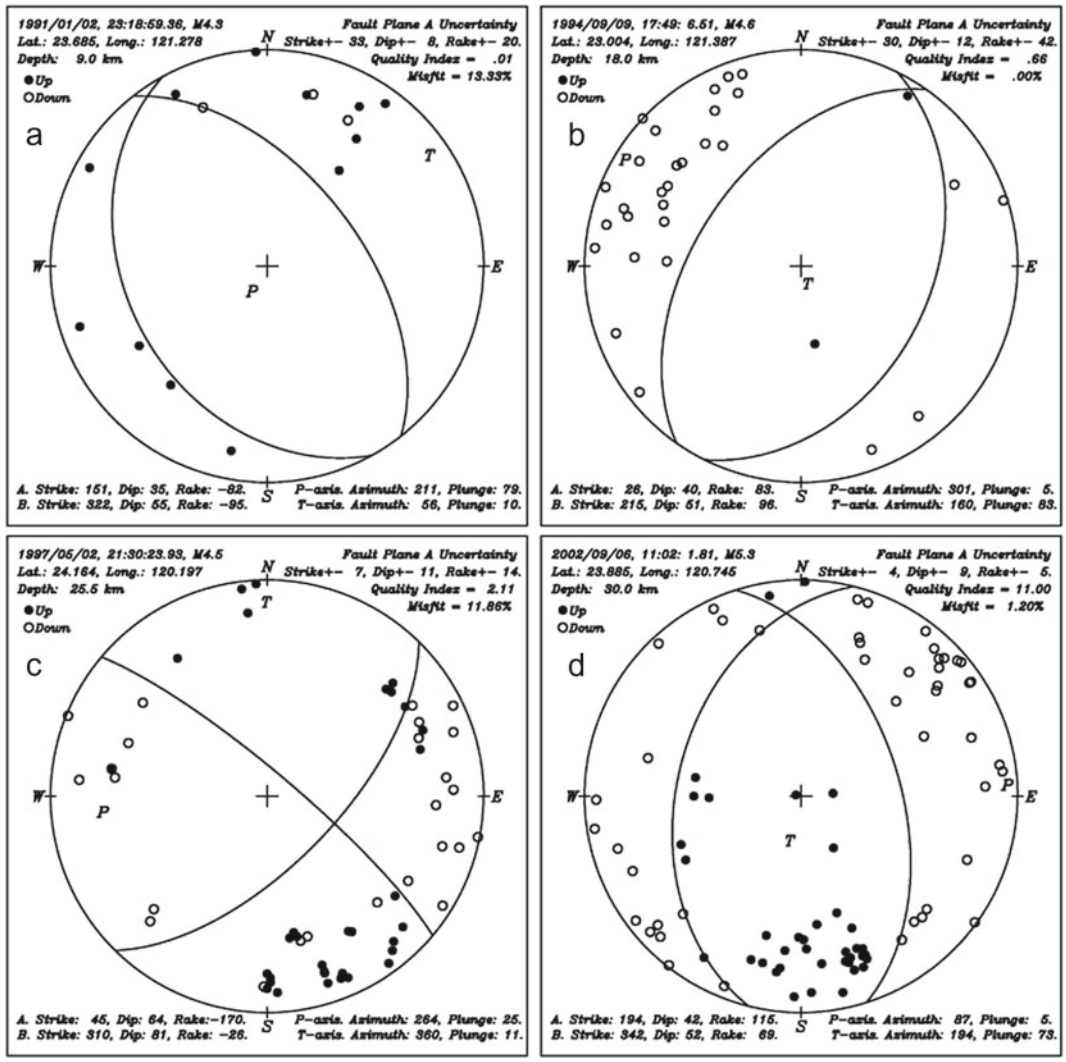


Fig. 26 Four focal-mechanism solutions with different quality indices. **a** A solution with a very low quality index value due to a large GAP, poor polarity fitness, and a small number of the dilatational first-motion readings; **b** a solution of a low-quality index value due to a small number of compressional first-motion readings; **c** a relatively poor polarity fitness results in a low-quality index value; **d** an example with a very good constraint (Wu et al. 2008)

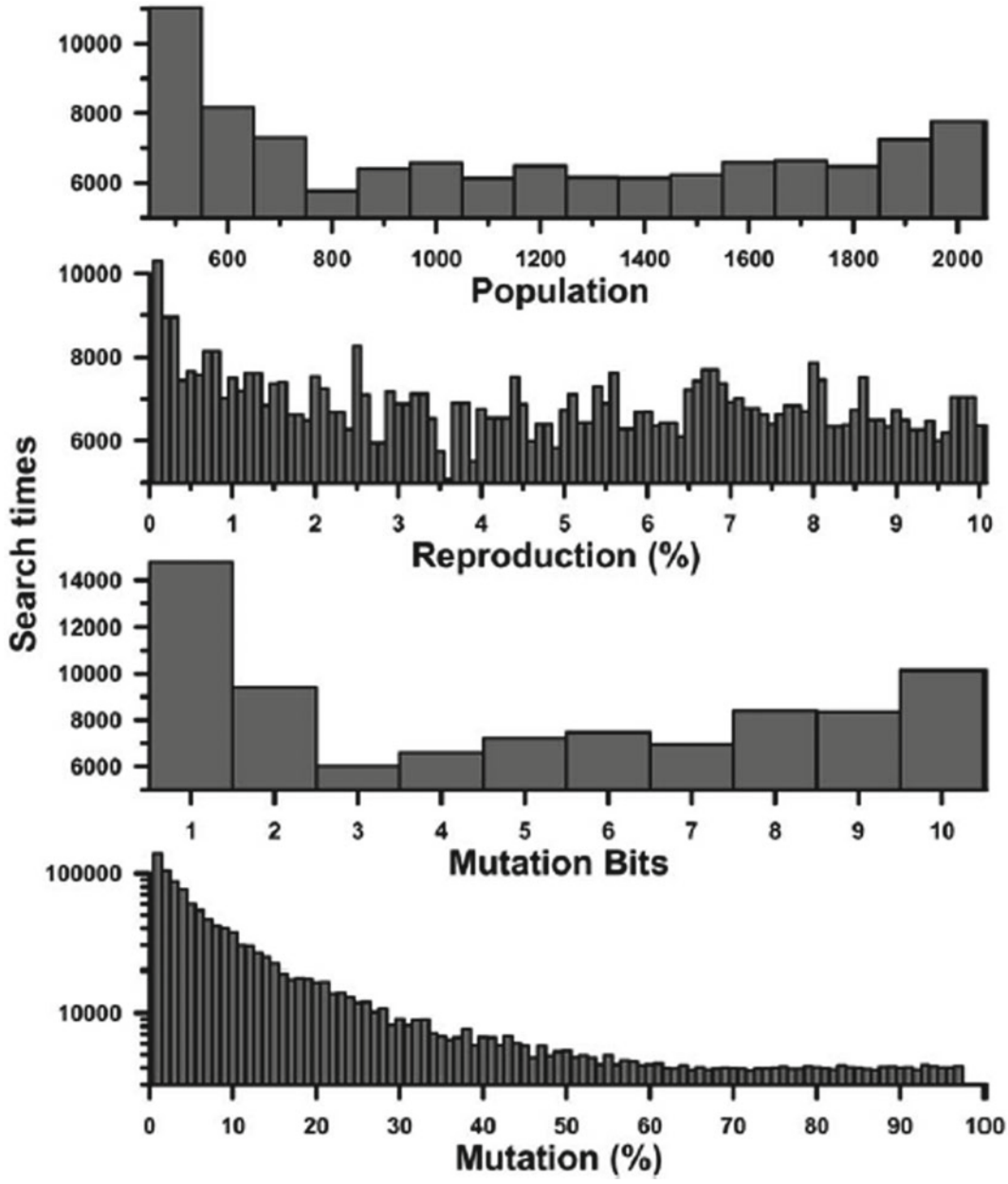


Fig. 27 Search time versus diverse settings of the GA parameters for synthetic data (Wu et al. 2008)

Table 21 The optimal parameters for GA-based focal mechanism estimator

Parameter	Best value
Population size	800
Reproduction rate	3.6%
Number of bits for reversing in mutation process	3 bits
Mutation rate	72%
Pure cross over rate	24.4%

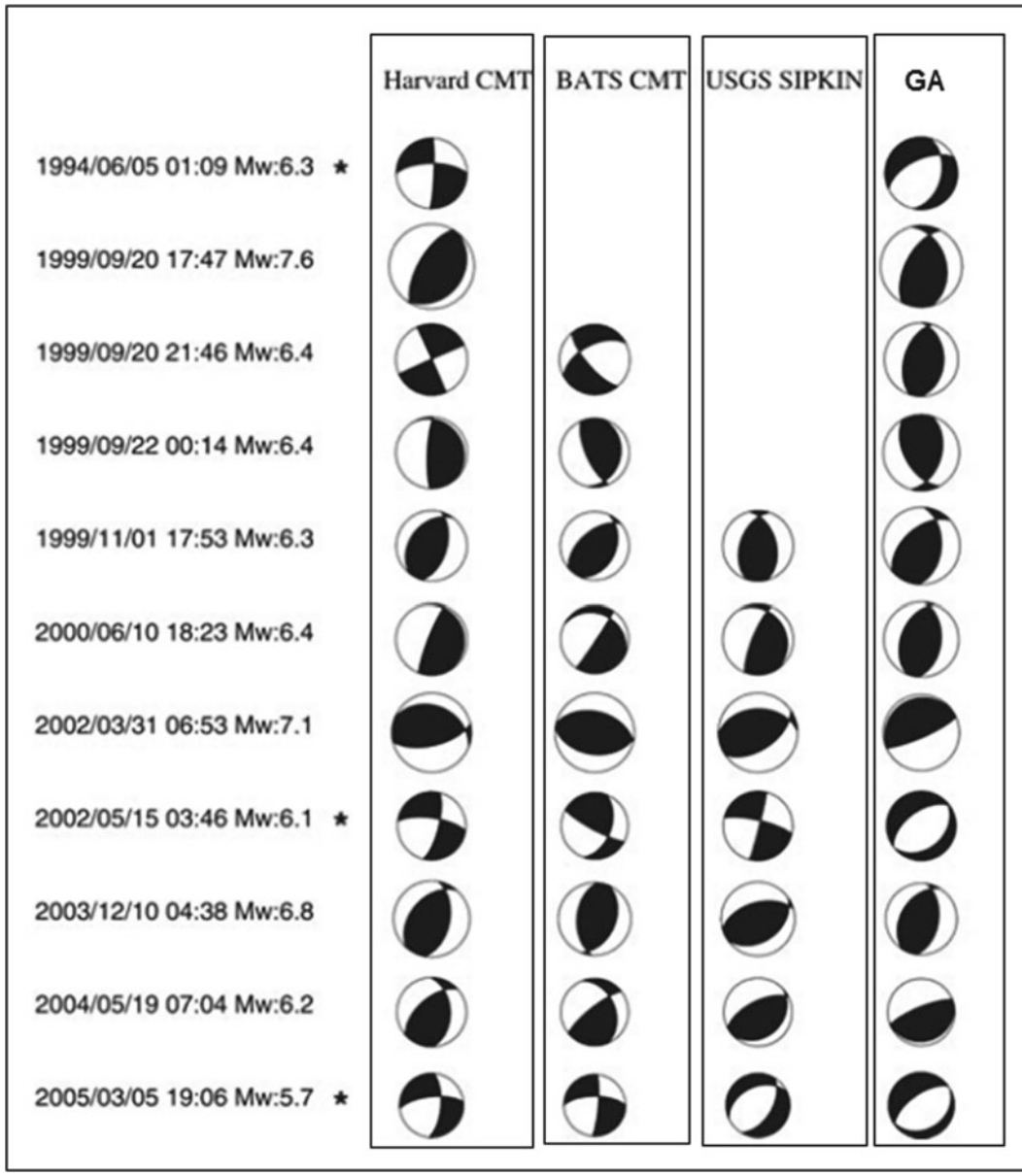


Fig. 28 Comparison of focal mechanisms determined through GA and the Harvard, USGS (Sipkin), and BATS solutions (redrawn after Wu et al. 2008)

References

Aloisi M, D'Agostino M, Kenneson GD, Mostaccio A, Neri G (2002) Satellite analysis and PUFF simulation of the eruptive cloud generated by Mt. Etna paroxysm of 22 July 1998. J Geophys Res 107:1–12

Aranha C, Lavinas YC, Ladeira M, Enescu B (2014) Is it possible to generate good earthquake risk models using genetic algorithms? In: Proceedings of the international conference on evolutionary computation theory and applications–ECTA, Rome, Italy, pp 49–58.

Aurnhammer M, Tönnies KD (2005) A genetic algorithm for automated horizon correlation across faults in seismic images. IEEE Trans Evol Comput 9(2):201–210

Bhattacharyya J, Sheehan AF, Tiampo KF, Rundle JB (1999) Using a genetic algorithm to model broadband regional waveforms for crustal structure in the Western United States. Bull Seismol Soc Am 89 (1):202–214

Böse M, Papadopoulos AN, Danciu L, Clinton JF, Wiemer S (2022) Loss-based performance assessment and seismic network optimization for earthquake early warning. Bull Seismol Soc Am. https://doi.org/10.1785/0120210298

Buckingham MJ, Garces MA (1996) A canonical model of volcano acoustics. J Geophys Res 101:8129–8151

Carbone D, Currenti G, Negro CD (2008) Multiobjective genetic algorithm inversion of ground deformation and gravity changes spanning the 1981 eruption of Etna volcano. J Geophys Res 113:1–10, B07406. https://doi.org/10.1029/2006JB004917

Currenti G, Del Negro C, Nunnari G (2005) Inverse modelling of volcanomagnetic fields using a genetic algorithm technique. Geophys J Int 163:403–418

Del Negro, C, Nunnari, G (2003) A software tool for modelling volcanomagnetic data. In: Proceedings of the international association for mathematica geology, IAMG 2003, Portsmouth, UK

Espinosa-Ramos JI, Vázquez RA (2011) Locating seismic-sense stations through genetic algorithm: genetic algorithms. In: GECCO '11: proceedings of the 13th annual conference on genetic and evolutionary computation, pp 941–948. https://doi.org/10.1145/2001576.2001705

Fitterman DV (1978) Electrokinetic and magnetic anomalies associated with dilatent regions in a layered earth. J Geophys Res 83:5923–5928

Fitterman DV (1981) Correction to theory of electrokinetic magnetic anomalies in a faulted half space. J Geophys Res. 86:9585– 9588

Garces MA, McNutt SR (1997) Theory of the airborne sound field generated in a resonant magma conduit. J Volcanol Geotherm Res 78:155–178

García S, Alcántara L (2019) Generator of genetic seismic signals. Geofísica Internacional 58–3:179–188

Ghaboussi J, Lin CJ (1998) New method of generating spectrum compatible accelerograms using neural networks. Earthq Eng Struct Dyn 27(4):377–396

Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading

Hagerty MT, Schwartz SY, Garces MA, Protti M (2000) Analysis of seismic and acoustic observations at Arenal Volcano, Costa Rica, 1995–1997. J Volcanol Geotherm Res 101:27–65

Hamano Y et al (1990) Geomagnetic variations observed after the 1986 eruption of Izu-Oshima volcano. J Geomagn Geoelectr 42:319–336

Kennett BLN, Sambridge MS (1992) Earthquake location genetic algorithms for teleseisms, physics of the earth and planetary interiors, 75, 103–110. Elsevier Science Publishers B V, Amsterdam

Kim Y, Ghaboussi J (1999) A new method of reduced-order feedback control using genetic algorithms. Earthq Eng Struct Dyn 28(3):235–254

Koper KD, Wysession ME, Wiens DA (1999) Multimodal function optimization with a niching genetic algorithm: a seismological example. Bull Seismol Soc Am 89(4):978–988

Li Y, Sui Q, Wang J, Wang Z, Jia L, Wang H, Du B (2017) Localization of microseismic source based on genetic-simplex hybrid algorithm. Chin Autom Congr (CAC) 2017:4002–4007. https://doi.org/10.1109/CAC.2017.8243480

Mathias Keith E, Whitley D, Storky C, Kusumayy T (1994) Staged hybrid genetic search for seismic data imaging. In: IEEE world congress on computational intelligence, Proceedings of the first IEEE conference on evolutionary computation. https://doi.org/10.1109/ICEC.1994.349925

Montalto P, Cannata A, Privitera E, Gresta S, Nunnari G, Patanè D (2010) Towards an automatic monitoring system of infrasonic events at Mt. Etna: strategies for source location and modeling. Pure Appl Geophys 167 (10):1215–1231. https://doi.org/10.1007/s00024-010-0051-y

Novianty A, Meilano I, Machbub C, Widiyantoro S, Susilo S (2021) Rapid estimation of earthquake magnitude and source parameters using genetic algorithms. Appl Sci 11(24):11852. https://doi.org/10.3390/app112411852

Nunnari G, Puglisi G, Guglielmino F (2005) Inversion of SAR data in active volcanic areas by optimization techniques. Nonlinear Process Geophys 12:863–870. SRef-ID: 1607-7946/npg/2005-12-863.

Sambride M, Mosegaard K (2002) Montecarlo methods in geophysical inverse problems. Rev Geophys 40(3). https://doi.org/10.1029/2000RG000089

Sociedad Mexicana de Ingeniería Sísmica (SMIS) (2000) Mexican strong motion database CD-Rom, vol 2

Sasai Y, Uyeshima M, Zlotnicki J, Utada H, Kagiyama T, Hashimoto T, Takahashi Y (2002) Magnetic and electric field observations during the 2000 activity of Miyake-jima volcano, Central Japan. Earth Planet Sci Lett 203:769–777

Tiampo KF, Fernandez J, Gentzsch G, Charco M, Rundle JB (2004) Inverting for the parameters of a volcanic source using a genetic algorithm and a model

for magmatic intrusion in elastic-gravitational layered Earth models. Comp Geosci 30(9-10):985–1001

Vergniolle S, Brandeis G (1994) Origin of the sound generated by Strombolian explosions. Geophys Res Lett 21:1959–1962

Vergniolle S, Brandeis G (1996) Strombolian explosions: a large bubble breaking at the surface of a lava column as a source of sound. J Geophys Res 101:20433–20448

Vergniolle S, Brandeis G, Mareschal JC (1996) Strombolian explosions: eruption dynamics determined from acoustic measurements. J Geophys Res 101:20449–20466

Vergniolle S, Boichu M, Caplan-Auerbach J (2004) Acoustic measurements of the 1999 basaltic eruption of Shishaldin volcano, Alaska: 1) Origin of Strombolian activity. J Volcanol Geotherm Res 137:109–134

Vergniolle S, Ripepe M (2008) From Strombolian explosions to fire fountains at Etna Volcano (Italy): what do we learn from acoustic measurements? Geol Soc London Spec Publ 307:103–124

Vergniolle S, Caplan-Auerbach J (2004) Acoustic measurements of the 1999 basaltic eruption of Shishaldin volcano, Alaska: 2) Precursor to the Subplinian activity. J Volcanol Geotherm Res 137:135–151

Wang F, Dai Y, Wang S (2009) Chaos-genetic algorithm based on the cat map and its application on seismic wavelet estimation. In: International workshop on chaos-fractals theories and applications, IEEE computer society, pp 112–116. https://doi.org/10.1109/IWCFTA.2009.31

Wu YM, Zhao L, Chang CH, Hsu YJ (2008) Focal-mechanism determination in Taiwan by genetic algorithm. Bull Seismol Soc Am 98(2):651–661. https://doi.org/10.1785/0120070115

Xiong T, Liu Z, Huang Z, Qing L, Xiong X (2011) The research on the joint application of the seismic attributes optimizing based on the genetic algorithm and the neural network methods. In: 7th International conference on natural computation, IEEE, pp 2336–2340. 978-1-4244-9953-3/11/$26.00 ©2011

Yilmaz S (2011) Ground motion predictive modelling based on genetic algorithms. Nat Hazards Earth Syst Sci 11:2781–2789. https://doi.org/10.5194/nhess-11-2781-2011

Zlotnicki J, Le Mouel JL (1988) Volcanomagnetic effects observed on Piton de la Fournaise Volcano (Reunion Island): 1985–1987. J Geophys Res. 93:9157–9171. https://doi.org/10.1029/JB093iB08p09157